Specification theories for probabilistic systems

Pedersen, Mikkel Larsen

Publication date:
2011

Document Version
Early version, also known as pre-print

Specification theories for probabilistic systems

Mikkel Larsen Pedersen
Department of Computer Science
Aalborg University, Denmark

A thesis submitted for the degree of PhD

October 2011

To Svanborg

Abstract

Today systems are too complex to be developed by a single team of developers, thus specifications of subsystems need to be supplied to subcontractors that work independently of each other. To avoid misunderstanding between designer and the subcontractor, specifications must be formal allowing to write specifications in a rigorous and precise manner, and it should be possible to check whether an implementation is a valid realization of a specification.

Throughout the thesis, we focus on systems exhibiting probabilistic behaviour. Probabilities are introduced in system design to model various aspects of real-life systems such as failures or uncertainties, e.g. the risk that a transmission of a message will fail may be estimated with a probability, and the uncertainty of a weather forecast may be represented by assigning probabilities to the possible outcome of the weather.

We advocate and present new results on specification theories for systems exhibiting both non-deterministic and probabilistic behavior. A specification theory is equipped with notions covering the relation between implementation and specification (satisfaction), the relation between specifications (refinement) together with composition operators that allow combining specifications (structural composition and logical composition).

Initially, we consider Markov Chains (MCs) as a basic model for representing probabilistic implementations. We investigate Interval Markov Chains (IMCs), as a specification theory for MCs. Whereas transitions of a given MC are taken with a specific probability, IMCs specify an interval of allowed probabilities. As IMCs are not closed under conjunction and parallel composition, we introduce the concept of Constraint Markov Chains (CMCs), that is the first complete specification theory for MCs.

We further explore the influence of non-deterministic behaviour by mixing CMCs and Modal Transition Systems, a well-known specification theory for Labelled Transition Systems, and consider Probabilistic Automata (PAs), that can be seen as extensions of MCs; in this model, state changes are additionally guarded by actions. We present a specification theory for PAs, namely Abstract Probabilistic Automata (APAs).

Finally, the tool APAC, that implements algorithms for CMCs and APAs, is introduced.

Dansk sammenfatning

I dag er systemer for komplekse til at blive udviklet af et enkelt hold af udviklere, og således er der behov for at specifikationer af delsystemer leveres til underleverandører, der arbejder uafhængigt af hinanden. For at undgå misforståelser mellem designer og underleverandør, skal specifikationer være formelle, hvilket tillader specifikationer skrevet på en stringent og præcis måde, og det skal være muligt at kontrollere, om en implementation er en gyldig realisering af en specifikation.

Igennem afhandlingen fokuserer vi på systemer, der har probabilistisk adfæ rd. Sandsynligheder bruges i systemdesign til at modellere forskellige aspekter af virkelige systemer såsom fejl eller usikkerheder, f.eks. kan risikoen for at en transmission af en besked mislykkes estimeres med en sandsynlighed, og usikkerheden på en vejrudsigt kan repræsenteres ved at tildele sandsynligheder til de mulige udfald af vejret.

Vi fremhæver og præsenterer nye resultater for specifikationsteorier for systemer der har både non-deterministisk og probabilistisk adfæ rd. En specifikationsteori er udstyret med begreber, der dækker forholdet mellem implementation og specifikation (tilfredsstillelse), forholdet mellem specifikationer (raffinering) samt sammensætningsoperatorer, som gør det muligt at kombinere specifikationer (strukturel sammensætning og logisk sammensætning).

Til at begynde med betragter vi Markov-kæder (MKer), som er en grundlæggende model der repræsenterer probabilistiske implementationer. Vi undersøger Interval-Markov-kæder (IMKer) som en specifikationsteori for MKer. Hvor transitioner i en given MK tages med en specifik sandsynlighed, angiver IMKer et interval af tilladte sandsynligheder. Da IMKer ikke er lukket under logisk og parallel sammensætning, introducerer vi begrebet Begræ nsnings-Markov-kæder (BMKer), som er den første komplette specifikationsteori for MKer.

Yderligere undersøger vi inflydelsen af non-deterministisk adfærd ved at kombinere BMKer og Modale Transitionssystemer, som er en velkendt specifikationsteori for Mæ rkede Transitionssystemer, og undersøger Probabilistiske Automater (PAer), der kan ses som udvidelse af MKer; i denne model er transitioner yderligere mæ rket med handlinger. Vi præsenterer en specifikationsteori for PAer, Abstrakte Probabilistiske Automater (APAer).

Endelig, bliver værktøjet APAC præsenteret, som implementerer algoritmer for BMKer og APAer.

Acknowledgements

I would like to thank my supervisor Kim G. Larsen for providing me with help and his neverending flow of good ideas.

An equally big thanks to my co-authors, and especially Axel Legay and Benoît Delahaye for our fruitful collaboration during their visits in Aalborg and my visits in Rennes. Also a big thanks to my office mates Line Juhl and Claus Thrane, Radu Mardare, and my former co-supervisor Jiří Srba, for nice discussions and good company.

Finally, a thanks to my family and friends, and especially my wife Svanborg, for coping with my absent-mindedness and stress around deadlines and my 3 month visit in Rennes.

Contents

Introduction

1 Quantitative specification theories

An embedded system is an engineering artifact involving computation that is subject to physical constraints [1]. The description defines mobile phones, mp3 players, washing machines, and microwave ovens etc. as embedded systems, but also safety-critical system such as braking systems in cars or GPS systems in planes etc. It is evident that we are increasingly dependent on embedded systems, so we need to eliminate or at least limit the amount and impact of errors in their building.

History reveals several examples of accidents related to bugs in embedded systems such as the failed Ariane 5 launch [2]. Bugs leading to accidents, such as this, are due to discrepancies that have been introduced in the process from requirements to implementation. The requirements of an embedded system fall into two categories [1]:

- Functional requirements: These specify the functionality and features of the system independent of how the implementation is derived.

- Extra-functional requirements: These specify performance requirements such as resource usage, failure rates, timing, power consumption etc.

When only considering functional requirements, one can only reason about the qualitative aspects of a system. However, in many systems, quantitative aspects are important e.g. that an airbag in a car is inflated at most 5 ms after an impact or that a server will consume at most 4000 kWh per year. The theme of this thesis will be the development of suitable specification theories for systems exhibiting probabilistic behaviour, which is a quantitative aspect.

Probabilities are introduced for several reasons. E.g. protocols such as Zigbee [3] and Firewire [4] use randomized behaviour and therefore call for specification theories for probabilistic systems. Also in modeling an unreliable system, probabilities may be used to estimate failure rates [1]. Knowing the probability for loosing a message in a transmission over an unreliable medium allows calculating the number of retransmissions for delivery guaranteed with probability 0.99, say. Also, one may not be able to avoid situations in which an undesirable event may occur, but being able to estimate the upper bound for the probability for getting in such situations, may prove useful.

In the following, we will motivate the requirements for a specification theory.

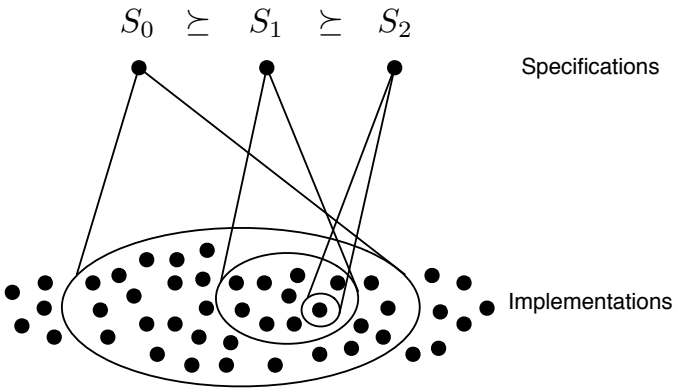


Figure 1: Stepwise refinement

Systems design and development

More and more companies building embedded system tend to rely on outsourcing for a number of reasons. They may want to benefit from the expertise of subcontractors, as well as exploit lower cost economies in foreign countries.

Typically, a company specifies the design and requirements of a system and high-level specifications for subcomponents are supplied to subcontractors that work independently of each other. The specification formalism must be rigorous and precise in the sense that there should be no misunderstandings between the designer and the subcontractor. To this end, one should focus on a mathematical framework and resort to formal methods in obtaining the ultimate goal for the subcontractor; that is, given that the specification is meaningful, to construct a subcomponent/implementation that conforms with the specification. The developer must be able to check that an implementation satisfies the specification. A well-known concept is that of *stepwise refinement*: The developer is given a high-level specification and refines it in one or more steps, until an implementation is reached, making choices in the design. If, in each step, the refinement is valid, then the implementation that is eventually reached will satisfy the initial specification. In each step, techniques such as model checking could be employed to assure validity.

Consider Figure 1 that illustrates the concept of stepwise refinement. The initial specification S_0 specifies a number of implementations illustrated by a circle. From S_0 a more concrete specification S_1 is constructed; it is more concrete as it only allows a subset of the implementations allowed by S_0. Finally S_2 is reached concretizing S_1. As it can be seen, S_2 only specifies a single implementation, so we have indeed reached an implementation.

Decomposing the high-level specification (see Figure 2) into specifications of subcomponents is a well-known principle known as *component-based design*. The rewards are:

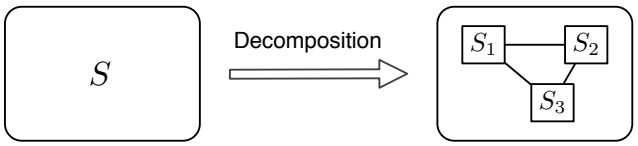


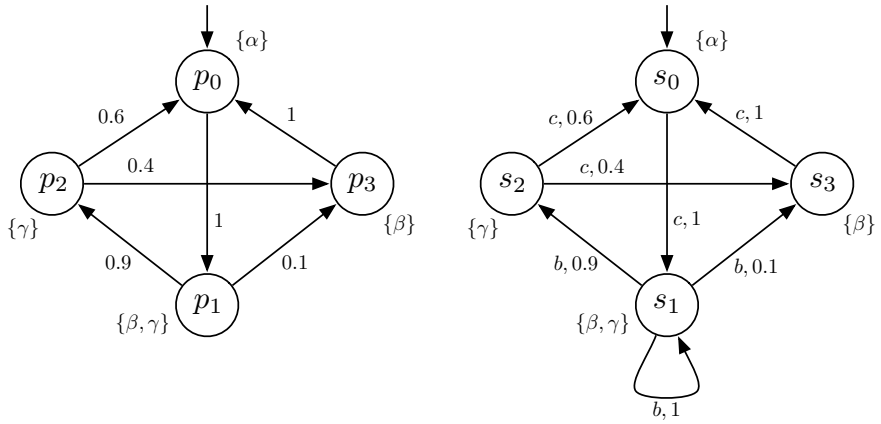
Figure 2: Decomposing the specification

1. Subcomponents may be implemented independently
2. Related functionalities can be grouped in a single subspecification
3. Each subspecification is less complex than the full specification
4. The separation may provide a better overview in terms of avoiding errors

But even though subcomponents may function correctly, there is a need for methods for reasoning about global properties, based on properties of subcomponents. Also, this reasoning is needed already at the specification level, since, e.g. for economical reasons, correctness must be assessed before the implementation is built.

To achieve the points advocated, the challenge is to develop suitable *specification theories*, encompassing the following points:

1. *Satisfaction.* The fundamental concept of a specification theory is satisfaction. It defines exactly when an implementation satisfies a specification.
2. *Consistency.* Given the notion of satisfaction, a specification is said to be consistent if it is satisfied by at least one implementation. Consistency is needed to verify that specification are well-formed and do not contain inconsistent or contradictory requirements.
3. *Refinement.* The notion of refinement is defined with respect to satisfaction i.e. a specification S refines another specification S', if all implementations satisfying S also satisfy S'. Refinement expresses correctness of a step-wise process, where more coarse-grained specifications are refined into more detailed ones, as illustrated in Figure 1.
4. *Conjunction (logical composition).* The conjunction operation should allow combining two specifications into a single specification that represents the intersection of sets of implementations of the two operands. This operation is important as it provides a method for deciding whether two specifications can be satisfied simultaneously, which would be the case, if the conjunction is consistent. The conjunction could be empty, if two requirements are contradictory.
5. *Parallel composition (structural composition).* The concept of parallel composition is needed to compose component specifications and should reflect the natural



(a) Markov Chain **(b)** A Probabilistic Automaton

Figure 3: The two basic probabilistic models

way of fitting together a number of subsystems to build a system. The parallel composition has to satisfy that the composition of two implementations satisfies the composition of their specifications. This is to make sure that the parallel composition of outsourced implementations of subcomponents will satisfy the specification of the whole system.

Ideally, the use of computers should be facilitated to establish whether an implementation satisfies a specification, whether a specification refines another specification, etc. This requires that establishing such relations is decidable. Even further, it is desirable that they can be checked efficiently.

Throughout this thesis, we will focus on specification theories with implementation being the two probabilistic models, Markov Chains (MCs) [5] and Probabilistic Automata (PAs) [6], as defined in Definitions 1 and 2, respectively.

Definition 1 (Markov Chain)**.** *A* Markov Chain *(MC) is a tuple* $C = (P, p_0, \pi, A, V_C)$*, where* P *is a set of states containing the initial state* p_0*,* A *is a finite set of atomic propositions,* $V_C : P \to 2^A$ *is a state valuation function, and* $\pi : P \to \text{Dist}(P)$ *is a probability distribution assignment such that* $\sum_{p' \in P} \pi(p)(p') = 1$ *for all* $p \in P$*.*

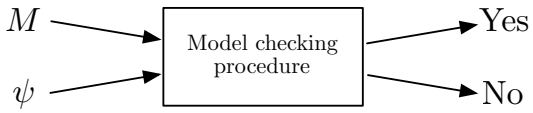


Figure 4: Model checking

Definition 2 (Probabilistic Automaton)**.** *A Probabilistic Automaton is a tuple* $N = (S, A, L, AP, V, s_0)$*, where* S *is the set of states containing the initial state* s_0*,* A *is a finite set of actions,* L*:* $S \times A \times \text{Dist}(S) \to \{\bot, \top\}$ *is a* two-*valued transition function,* AP *is a finite set of atomic propositions, and* V*:* $S \to 2^{AP}$ *is a state-labeling function.*

We will begin by assuming state sets to be countable. In the thesis contribution (Section 4 and onwards), however, we will moreover require them to be finite. Figures 3a and 3b illustrate a MC and a PA, respectively.

2 Logical and process algebraic specification theories

In this section, some logical and process algebraic views on specification theories in relation to Markov Chains and Probabilistic Automata are presented. At the end of the section, their forces and shortcomings will be discussed.

2.1 Logical approach

Satisfaction checking for logical specifications is known as model checking, and refers to the process of, given a model M of a system and a logical formula ψ, to determine whether M satisfies ψ. The discipline is illustrated in Figure 4.

The term 'model checking' was coined by Clarke and Emerson [7]. Today, model checking is a widely used concept and is supported by tools such as UPPAAL [8], SMV [9], SPIN [10], and PRISM [11]. The concept of quantitative model checking [12] has arisen to support quantitative answer to model checking e.g. that a property is satisfied with a probability p.

Temporal logics, i.e. logics for reasoning about system behaviour over time, was proposed in [13] for specifying requirements. Famous temporal logics are Linear Temporal Logic (LTL) [13] and Computation Tree Logic (CTL) [14], both specifying state-labelled transition systems [15]. While LTL formulas specify properties on every path from a given state s, CTL formulas specify properties on the computation tree starting in s using path quantifier $\exists$ and $\forall$ to distinguish whether properties should hold in some or

all paths, respectively. The logics are incomparable i.e. there exists CTL formulas that can not be expressed in LTL and vice versa.

Let us review a probabilistic extension of CTL. Other probabilistic logics are e.g. Probabilistic Modal Logic [16] and Continuous Stochastic Logic [17].

Probabilistic Computation Tree Logic

Probabilistic Computation Tree Logic (PCTL) [18] is a probabilistic extension of CTL, making it possible to reason about so-called soft deadlines i.e. whether a property P holds with a certain probability at least p within t time units. Formulae are interpreted over MCs (as defined in Definition 1 and will be treated again in Section 4.1).

The syntax of PCTL is as follows:

Definition 3 (PCTL syntax)**.**

$$\begin{array}{lrcl} [\textit{state formulae}] & \psi & ::= & a \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \psi \rightarrow \psi \mid [\varphi]_{\triangleright p} \\ [\textit{path formulae}] & \varphi & ::= & \psi U^{\leq t} \psi \mid \psi \mathcal{U}^{\leq t} \psi, \end{array}$$

where $t \in \mathbb{N} \cup \{\infty\}$*,* $p \in [0,1]$*,* $a \in A$*, and* $\triangleright \in \{>, \geq\}$*.*

Intuitively, we say that a state s satisfies $[\varphi]_{\triangleright p}$ if and only if φ holds on a path starting at s with probability $\triangleright p$. We say that $\psi_1 U^{\leq t} \psi_2$ if and only if ψ_2 becomes true after at most t time units and that ψ_1 remains true until that point. The path formula $\psi_1 \mathcal{U}^{\leq t} \psi_2$ weakens the former by saying that either ψ_1 remains true for at least t time units or $\psi_1 U^{\leq t} \psi_2$.

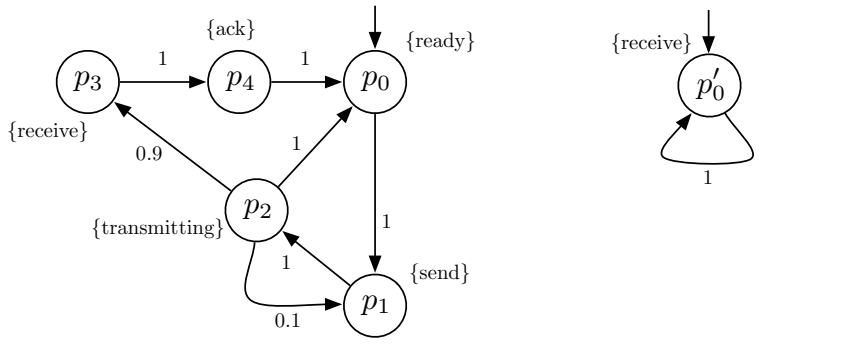
Given a state s_0 of a MC $C = (P, p_0, \pi, A, V_C)$, we define a path σ as an infinite sequence $\sigma = s_0 \rightarrow s_1 \cdots \rightarrow s_n \rightarrow \cdots$ of states in P. We denote the $(n+1)$'th state in σ as $\sigma[n]$, that is, $s_n = \sigma[n]$, and the prefix of σ of length $n+1$ is denoted by $\sigma \uparrow n$, that is, $\sigma \uparrow n = s_0 \rightarrow s_1 \cdots \rightarrow s_n$.

Consider the probability space $(X, \mathcal{A})$ where X is the set of paths starting in s_0 and $\mathcal{A}$ is a σ-algebra on X generated by sets $\{\sigma \in X | \sigma \uparrow n = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n\}$ ranging over all finite paths $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$. The measure μ_m is uniquely defined as, for every finite path $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$,

$$\mu_m\left(\{\sigma \in X | \sigma \uparrow n = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n\}\right) = \pi(s_0)(s_1)\pi(s_1)(s_2)\cdots\pi(s_{n-1})(s_n),$$

with the definition that $\mu_m\left(\{\sigma \in X | \sigma \uparrow 0 = s_0\}\right) = 1$.

Given a formula ψ, the satisfaction relation $\models_C$ between a MC $C = (P, p_0, \pi, A, V_C)$ and ψ is inductively given in the semantic of the logic [18] as follows.



(a) A protocol **(b)** A trivial implementation

Figure 5: Two protocol implementations

$$
\begin{aligned}
s \models_C a &\quad\Leftrightarrow\quad a \in V_C(s)\\
s \models_C \neg\psi &\quad\Leftrightarrow\quad \neg(s \models_C \psi)\\
s \models_C \psi_1 \wedge \psi_2 &\quad\Leftrightarrow\quad s \models_C \psi_1 \wedge s \models_C \psi_2\\
s \models_C \psi_1 \vee \psi_2 &\quad\Leftrightarrow\quad s \models_C \psi_1 \vee s \models_C \psi_2\\
s \models_C \psi_1 \rightarrow \psi_2 &\quad\Leftrightarrow\quad s \models_C \neg\psi_1 \vee s \models_C \psi_2\\
s \models_C [\psi]_{\triangleright p} &\quad\Leftrightarrow\quad \mu_m(\{\sigma \in X | \sigma[0] = s \wedge \sigma \models'_C \psi\}) \triangleright p\\
\sigma \models'_C \psi_1 U^{\leq t} \psi_2 &\quad\Leftrightarrow\quad \exists i \leq t : \sigma[i] \models_C \psi_2 \wedge \forall 0 \leq j < i : \sigma[j] \models_C \psi_1\\
\sigma \models'_C \psi_1 \mathcal{U}^{\leq t} \psi_2 &\quad\Leftrightarrow\quad \forall 0 \leq j \leq t : \sigma[j] \models_C \psi_1 \vee \sigma \models'_C \psi_1 U^{\leq t} \psi_2
\end{aligned}
$$

We say that C satisfies ψ if and only if $p_0 \models_C \psi$. The set $\{\sigma \in X | \sigma[0] = s \wedge \sigma \models'_C \psi\}$ is measurable as it can be constructed as a countable union of elements in the σ-algebra $\mathcal{A}$ [15].

Well-known CTL formulae such as $AG\psi$ and $AF\psi$ can be expressed in PCTL as $\left[\psi\ \mathcal{U}^{\leq\infty}\mathsf{false}\right]_{\geq 1}$ and $\left[\mathsf{true}\ \mathcal{U}^{\leq\infty}\psi\right]_{\geq 1}$, respectively.

Example 1. *This example illustrates the satisfaction relation of PCTL. Consider the protocol [18] illustrated in Fig. 5a. After the initial state, a message is sent over an unreliable medium; the transmission might fail (state p_2). Upon reception (state p_3), an acknowledgement is sent to the sender (state p_4). We assume that acknowledgements are not lost.*

We want to specify that, after a `send`*, a* `receive` *is reached within 5 time units with a probability at least* 0.99*. This is specified in Equation* (1)*. Following the computations of [18], this requirement is satisfied by the MC in Fig. 5a.*

$$\varphi = \left[\left(send \rightarrow \left[\mathit{true}\ U^{\leq 6} receive\right]_{\geq 0.99}\right) \mathcal{U}^{\leq\infty}\mathit{false}\right]_{\geq 1} \tag{1}$$

Notice that the specification is also satisfied by the implementation in Fig. 5b.

2.2 Process algebraic approach

The notion of process algebras is based on the view that the behaviour of a reactive system, often viewed as a Labelled Transition System (LTS) [19], can be seen as a set of processes executing in parallel while communicating. A process algebra provides

- the representation of processes as terms S, T, etc.,
- essential operations [20] such as choice i.e. $S + T$ continues as S or T, sequential composition i.e. $S;T$ performs S and then T, and parallel composition i.e. $S \mid T$ represents S and T running in parallel, to construct processes from existing ones,
- an operational semantics that gives the behaviour of processes as e.g. a LTS, and
- behavioral equivalences, such as bisimulation [21, 22] or probabilistic bisimulation [16], that relates processes if they behave similarly.

A process algebra may act as a specification theory for reactive system with specifications and implementations being described in the same language. For defining satisfaction, we say that a process S satisfies another process T if and only if S and T are equivalent e.g. if S is (probabilistically) bisimilar T.

Some of the most prominent and well-known examples of process algebras [23] are Calculus of Communication Systems (CCS) [24], Communicating Sequential Processes [25], and Algebra of Communicating Processes [26]. Informally (and not exhaustive), in CCS, a process P can be written as the process $a.P_1$ that can perform an a-action and become P_1 i.e. $a.P_1 \xrightarrow{a} P_1$, the sum of process $\sum_{i \in I} P_i$, where I is an index set, and the parallel composition $P_1 \mid P_2$.

The operational semantics is given as a LTS, and CCS features various notions of equivalences, e.g. bisimulation, for conformance checking between processes. Bisimilar processes can match each others transition reaching bisimilar processes.

Definition 4. *An relation $\mathcal{R}$ on $\mathcal{P}$ is a (strong) bisimulation relation if and only if, whenever $P_1 \,\mathcal{R}\, P_2$, then*

- *for all $a \in Act$, if $P_1 \xrightarrow{a} P_1'$, then $P_2 \xrightarrow{a} P_2'$ and $P_1' \,\mathcal{R}\, P_2'$, and*
- *for all $a \in Act$, if $P_2 \xrightarrow{a} P_2'$, then $P_1 \xrightarrow{a} P_1'$ and $P_1' \,\mathcal{R}\, P_2'$.*

Concurrency Workbench [27] provides tool support for bisimulation checking, finding a distinguishing formula in Hennessy-Milner logic if two processes are not bisimilar, etc.

We will now review a probabilistic extension of CCS. Other probabilistic process algebras are e.g. Timed Probabilistic Calculus for Communicating Systems [28] and ACP$_\pi^+$ [29].

Probabilistic CCS

Probabilistic CCS (PCCS) [30, 31] is a probabilistic extension of CCS. Let $\mathcal{A}$ be a set of actions, let τ be a special action not in $\mathcal{A}$, let $Act = \mathcal{A} \cup \{\overline{a} | a \in \mathcal{A}\} \cup \{\tau\}$, and let $\mathcal{K}$ be a set of process names. The syntax of PCCS is as follows.

$$\frac{}{a.\bigoplus_{i\in I} p_i P_i \xrightarrow{a} \pi}\ \pi(P) = \textstyle\sum_{i\in I,\ P_i = P} p_i \qquad \frac{P_i \xrightarrow{a} \pi}{\sum_{i\in I} P_i \xrightarrow{a} \pi}\ i \in I$$

$$\frac{P_1 \xrightarrow{a} \pi_1}{P_1 \mid P_2 \xrightarrow{a} \pi_1 \mid \lambda P_2.1} \qquad \frac{P_2 \xrightarrow{a} \pi_2}{P_1 \mid P_2 \xrightarrow{a} \lambda P_1.1 \mid \pi_2}$$

$$\frac{P_1 \xrightarrow{a} \pi_1 \quad P_2 \xrightarrow{\overline{a}} \pi_2}{P_1 \mid P_2 \xrightarrow{\tau} \pi_1 \mid \pi_2} \qquad \frac{P_1 \xrightarrow{\overline{a}} \pi_1 \quad P_2 \xrightarrow{a} \pi_2}{P_1 \mid P_2 \xrightarrow{\tau} \pi_1 \mid \pi_2}$$

$$\frac{P \xrightarrow{a} \pi}{P \setminus A \xrightarrow{a} \pi \setminus A}\ a, \overline{a} \notin A \qquad \frac{P \xrightarrow{a} \pi \quad f(a) = b}{P[f] \xrightarrow{b} \pi[f]}$$

$$\frac{P \xrightarrow{a} \pi}{K \xrightarrow{a} \pi}\ K \stackrel{\text{def}}{=} P$$

Table 1: Operational semantics for PCCS (transitions)

Definition 5 (PCCS syntax)**.** *The collection of PCCS processes $\mathcal{P}$ is given by the following grammar*

$$P \quad ::= \quad a.\bigoplus_{i\in I} p_i P_i \ \mid\ \sum_{i\in I} P_i \ \mid\ P_1 \mid P_2 \ \mid\ P \setminus A \ \mid\ P[f] \ \mid\ K,$$

where $a \in Act$, I is an index set, $A \subseteq Act \setminus \{\tau\}$, $f : Act \to Act$ is a renaming function satisfying that $f(\tau) = \tau$ and $\forall a \in Act \setminus \{\tau\}, f(\overline{a}) = \overline{f(a)}$, and $K \in \mathcal{K}$.

The behaviour of each process name K is defined to be that of a process $P \in \mathcal{P}$, denoted by $K \stackrel{\text{def}}{=} P$.

In PCCS, action prefixing (as seen above for CCS) is substituted with a probabilistic choice guarded by an action,

$$a.\bigoplus_{i\in I} p_i \cdot P_i,$$

where a is an action, and for all $i \in I$, $p_i \in]0, 1]$, and $\sum_{i\in I} p_i = 1$. This process first performs the action a, and then becomes P_i for some $i \in I$ according to the induced probability distribution. If $I = \{1, \ldots, n\}$, we write $a.\bigoplus_{i\in I} p_i \cdot P_i = a.(p_1 \cdot P_1 \oplus \ldots \oplus p_n \cdot P_n)$. Action prefixing is obtained by letting I be a singleton i.e. $a.P_1$ is obtained as $a.\bigoplus_{i\in\{1\}} p_i \cdot P_i = a.1 \cdot P_1$.

The operational semantics of PCSS [30, 31] is given as a Probabilistic Automaton. Remark that we here use a definition of a PA that does not introduce the state labeling function as in Definition 2.

Given an expression P, we construct the PA $N = (S, A, L, s_0)$ where S are the processes reachable from P, $A = Act$, $s_0 = P$, and L is given as the transitions and operations on distributions in Table 1 and 2 with the rule that $L(s, a, \pi) = \top$ if and only if $s \xrightarrow{a} \pi$.

Extending CCS, PCCS is equipped with probabilistic bisimulation [16].

Definition 6. *An equivalence relation $\mathcal{R}$ on $\mathcal{P}$ is a probabilistic bisimulation if and only if, whenever $P_1 \mathcal{R} P_2$ and $a \in Act$, then*

- *if $P_1 \xrightarrow{a} \pi_1$, then $P_2 \xrightarrow{a} \pi_2$ and $\forall S \in \mathcal{P}/\mathcal{R} : \sum_{s \in S} \pi_1(s) = \sum_{s \in S} \pi_2(s)$.*

Example 2. *The specification of Example 1 is also expressible in PCCS. However, there will be a few differences due to the following reasons:*

- *PCCS only supports fixed probabilities, that is, we can not express that a property should hold with some probability bound.*
- *In PCCS, the language of implementation and specifications are the same.*
- *PCCS does not support the concept of state valuations.*

The specification in PCCS will be the process P. It is probabilistically bisimilar to P', and hence, P satisfies P'.

$$
\begin{array}{lcl|lcl}
P & \stackrel{\text{def}}{=} & \mathit{initialization}.1 \cdot P_1 & P' & \stackrel{\text{def}}{=} & \mathit{initialization}.1 \cdot P'_1 \\
P_1 & \stackrel{\text{def}}{=} & \mathit{send}.1 \cdot P_2 & P'_1 & \stackrel{\text{def}}{=} & \mathit{send}.1 \cdot P'_2 \\
P_2 & \stackrel{\text{def}}{=} & \mathit{transmission}. & P'_2 & \stackrel{\text{def}}{=} & \mathit{transmission}. \\
 & & (0.1 \cdot P_1 \oplus 0.9 \cdot P_3) & & & \left(0.1 \cdot P'_1 \oplus 0.4 \cdot P'_{3,1} \oplus 0.5 \cdot P'_{3,2}\right) \\
P_3 & \stackrel{\text{def}}{=} & \mathit{receive}.1 \cdot P_4 & P'_{3,1} & \stackrel{\text{def}}{=} & \mathit{receive}.1 \cdot P'_4 \\
 & & & P'_{3,2} & \stackrel{\text{def}}{=} & \mathit{receive}.1 \cdot P'_4 \\
P_4 & \stackrel{\text{def}}{=} & \mathit{ack}.1 \cdot P_1 & P'_4 & \stackrel{\text{def}}{=} & \mathit{ack}.1 \cdot P'_1
\end{array}
$$

2.3 Discussion

Certain specifications seem easier to express in a process algebra than in a logic and vice versa. While the essence of a process algebra is to reason about operational requirements such as the occurrence of specific actions in certain states, this can be difficult to express in logics. Logics, on the other hand, seem superior for expressing declarative specifications such as "if the button is pressed, then eventually coffee will be served".

As compositionality is a crucial property of a specification theory, probabilistic bisimulation is a valid choice for defining satisfaction for probabilistic process algebras as it enjoys useful properties e.g. for PCCS, it is a congruence [32]. However, with an equivalence defining satisfaction, the set of implementations of a process algebraic specification is always just a single equivalence class. This is not desirable, as a specification should possibly represent a variety of implementations, that are not necessarily equivalent. Also, refinement (Figure 1), characterized by implementation set inclusion, will reduce to equality between sets. Conjunction is also uninteresting in this case, as it is empty if the operands are not equivalent, and if non-empty, it will be equivalent to both operands.

$$(\pi_1 \mid \pi_2)(P) := \begin{cases} \pi_1(P_1)\pi_2(P_2) & \text{if } P = P_1 \mid P_2 \\ 0 & \text{else} \end{cases}$$

$$(\pi \setminus A)(P) := \begin{cases} \pi(P') & \text{if } P = P' \setminus A \\ 0 & \text{else} \end{cases}$$

$$(\pi[f])(P) := \begin{cases} \pi(P') & \text{if } P = P'[f] \\ 0 & \text{else} \end{cases}$$

Table 2: Operational semantics for PCCS (operations on distribution)

For logics, the notion of satisfaction between a formula and the underlying semantic model is given through the semantic rules. This is a more meaningful notion of satisfaction, as it allows implementations that are not necessarily equivalent. As an example, the two MCs in Figure 5a and 5b are not probabilistically bisimilar (using the definition of [15]), but they both satisfy the logical specification φ of Equation (1). However, there will exist a PCTL formula distinguishing them.

On the other hand, logics, such as PCTL, do not possess an operator for composing formulae in parallel. However, a natural definition, similar to that of Hennessy-Milner Logic in [33], is the operator $|$ with the semantics given as,

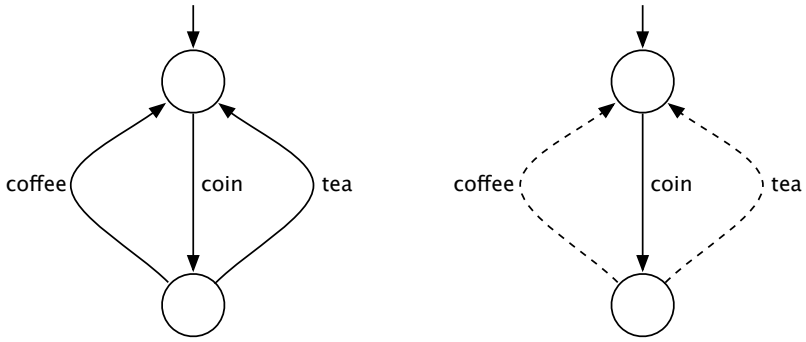
$$P \models \psi_1|\psi_2 \iff \exists P_1, P_2 : P \equiv P_1|P_2 \wedge P_1 \models \psi_1 \wedge P_2 \models \psi_2,$$

for an appropriate definition of $\equiv$. This definition, however, does not provide any methods for "evaluating" the symbolic representation $\psi_1|\psi_2$ i.e. constructing a formula ψ from $\psi_1|\psi_2$ such that $\forall P : P \models \psi \Leftrightarrow P \models \psi_1|\psi_2$. We consider this as a drawback, as it is desirable to be able to calculate the parallel composition explicitly. In [34], Modular Markovian Logic (with semantics given as Modular Markov Processes) is presented along with a complete axiomatization including rules for the expansion of the parallel compositition operator. However, decidability results have not been considered.

In the following section, we will consider an extension of Labelled Transition Systems, namely Modal Transition Systems. Modal Transition Systems, as we will see, support both parallel composition and conjunction, and features meaningful notions of satisfaction and refinement.

3 Modal Transition Systems

The notion of a Modal Transition System (MTS) was first presented in [35] as a specification theory for Labelled Transition Systems and the formalism possesses the desirable properties [36] for a specification theory. MTSs position themselves between two other specification theories for LTSs, namely, the process algebra Calculus of Communicating Systems and Hennessy-Milner Logic (HML), in the following way:



(a) A Labelled Transition System **(b)** A Modal Transition System

Figure 6: A LTS and a MTS

- MTSs support parallel composition and conjunction.
- In terms of expressivity, it has been shown in [37] that every MTS can be transformed into a HML formula, but the reverse does not hold, and every CCS expression can be transformed to a (trivial) MTS having the same set of implementations.

A LTS is formally defined as follows:

Definition 7. *A Labelled Transition System (LTS) L over an action alphabet A is a tuple $L = (P, \longrightarrow, p_0)$, where P is a set of states, p_0 the initial state, and $\longrightarrow \subseteq P \times A \times P$ are the transitions.*

The standard representation is a directed graph, as can be seen in Figure 6a. This transition system illustrates the behaviour of a simple coffee machine. Upon insertion of a coin, coffee or tea can be produced.

In [35], the authors argue that the use of process algebra for specifying LTSs is too weak, as they do not allow for *underspecificed* behaviour to be modelled. A MTS is a specification theory for LTSs that allows for such underspecified behaviour.

Definition 8. *A Modal Transition System (MTS) M over an action alphabet A is a tuple $M = (P, \longrightarrow, \dashrightarrow, p_0)$, where P is a set of states, p_0 the initial state, and $\longrightarrow \subseteq \dashrightarrow \subseteq P \times A \times P$ are must and may transitions, respectively.*

For an action $a \in A$ and states $p, p' \in P$, we write $p \overset{a}{\dashrightarrow} p'$ and $p \overset{a}{\longrightarrow} p'$ if $(p, a, p') \in \dashrightarrow$ and $(p, a, p') \in \longrightarrow$, respectively. If $(p, a, p') \notin \dashrightarrow$ for any p', we write $p \not\overset{a}{\dashrightarrow}$.

By deeming transitions as either may or must transition, it is specified which transitions are required (must) and allowed (may) in an implementation. Notice that all required transitions are also allowed. A MTS is displayed in Figure 6b.

Recently the notion of MTS has been extended with weight intervals on transitions [38] to facilitate modeling of e.g. resource usage, as well as modal specification with an explicit presentation of data [39]. Parametric Modal Transition Systems [40], with similarities to Acceptance Specifications [41], have been introduced as extensions of MTSs, where parameters are used to describe exclusive, conditional and persistent choices between transitions.

3.1 Refinement

The satisfaction between LTSs and MTSs is established through the concept of modal refinement. As a LTS is a MTS with only required transitions, i.e. $\longrightarrow = \dashrightarrow$, modal refinement is also defining a relationship between MTSs. As is turns out, modal refinement approximates refinement by implementation set inclusion, which we from now on will call thorough refinement. We will touch on this subject again later.

Definition 9 (Modal refinement)**.** *Let $M_1 = (P_1, \longrightarrow_1, \dashrightarrow_1, p_0^1)$ and $M_2 = (P_2, \longrightarrow_2, \dashrightarrow_2, p_0^2)$ be MTSs over alphabet A. A relation $\mathcal{R} \subseteq P_1 \times P_2$ is a modal refinement relation if and only if, whenever $(p_1, p_2) \in \mathcal{R}$, then*

1. *for all $a \in A$, if $p_2 \xrightarrow{a}_2 p_2'$, then $p_1 \xrightarrow{a}_1 p_1'$ and $(p_1', p_2') \in \mathcal{R}$, and*
2. *for all $a \in A$, if $p_1 \overset{a}{\dashrightarrow}_1 p_1'$, then $p_2 \overset{a}{\dashrightarrow}_2 p_2'$ and $(p_1', p_2') \in \mathcal{R}$.*

We say that M_1 (modally) refines M_2, written $M_1 \preceq_m M_2$, if and only if there exists a modal refinement relation $\mathcal{R} \subseteq P_1 \times P_2$ with $(p_0^1, p_0^2) \in \mathcal{R}$. Given a MTS M, we denote by $[\![M]\!]$, the set of LTS that refines M, that is, $[\![M]\!] = \{L \mid L \text{ is a LTS} \wedge L \preceq_m M\}$.

If a transition is required in the MTS M_2, it must also be required in M_1, and an allowed transition in M_1 must also be allowed in M_2. Informally, going from M_2 to M_1, an allowed, but not required, transition in M_2 can be 1) removed in M_1, 2) allowed in M_1, or 3) required in M_1. A required transition in M_2, however, must be present in M_1. If both M_1 and M_2 are LTSs, then modal refinement coincides with bisimulation as defined in Definition 4.

As an example, it is easy to see that the LTS in Figure 6a satisfies the MTS in Figure 6b. However, for a coffee drinking customer, it may seem odd that a coin is required, but coffee is not guaranteed. He may want to propose the MTS is Figure 7. This MTS is a refinement of that in Figure 6b.

The papers [35, 42] also motivate MTSs by the concept of stepwise refinement; from the initial specification of a system S_0, several refined specifications are developed before reaching an implementation I:

$$S_0 \succeq S_1 \succeq \cdots \succeq S_k \succeq I. \tag{2}$$

Stepwise refinement is not tied to a specific notion of refinement. Figure 1 illustrates stepwise refinement with respect to thorough refinement.

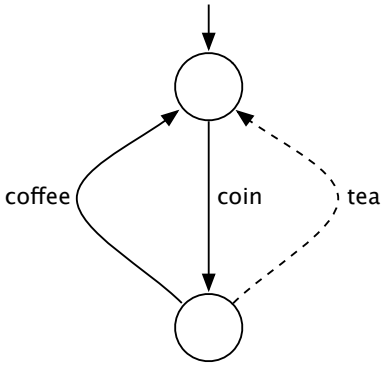


Figure 7: Another modal transition system

Definition 10. *Let* $M_1 = (P_1, \longrightarrow_1, \dashrightarrow_1, p_0^1)$ *and* $M_2 = (P_2, \longrightarrow_2, \dashrightarrow_2, p_0^2)$ *be MTSs. We say that* M_1 *thoroughly refines* M_2*, denoted* $M_1 \preceq_{th} M_2$*, if and only if* $[\![M_1]\!] \subseteq [\![M_2]\!]$.

Modal refinement approximates thorough refinement in the sense that modal refinement implies thorough refinement, but the other direction only holds for deterministic MTSs. In other words, modal refinement is sound, but not complete with respect to thorough refinement. Recently, in [43], it was shown that thorough refinement checking is EXPTIME-complete, while modal refinement is decidable in polynomial time.

3.2 Compositional operators

MTS support parallel composition and conjunction [41]. The former will be treated first.

Definition 11. *Let* $M_1 = (P_1, \longrightarrow_1, \dashrightarrow_1, p_0^1)$ *and* $M_2 = (P_2, \longrightarrow_2, \dashrightarrow_2, p_0^2)$ *be MTSs over the same alphabet. The parallel composition of* M_1 *and* M_2 *is the MTS* $M_1 \| M_2 = (P_1 \times P_2, \longrightarrow, \dashrightarrow, (p_0^1, p_0^2))$*, that is defined as follows: for each* $(p_1, p_2) \in P_1 \times P_2$*,*

$$\frac{p_1 \xrightarrow{a}_1 p_1' \quad p_2 \xrightarrow{a}_2 p_2'}{(p_1,p_2) \xrightarrow{a} (p_1',p_2')} \qquad \frac{p_1 \xdashrightarrow{a}_1 p_1' \quad p_2 \xdashrightarrow{a}_2 p_2'}{(p_1,p_2) \xdashrightarrow{a} (p_1',p_2')}$$

$$\frac{p_1 \xdashrightarrow{a}_1 p_1' \quad p_2 \not\xdashrightarrow{a}_2}{(p_1,p_2) \not\xdashrightarrow{a}} \qquad \frac{p_1 \not\xdashrightarrow{a}_1 \quad p_2 \xdashrightarrow{a}_2 p_2'}{(p_1,p_2) \not\xdashrightarrow{a}}$$

The parallel composition operator has the property that, given MTSs M_1, M_2, M_3, and M_4 such that $M_1 \preceq_m M_2$ and $M_3 \preceq_m M_4$, we have that $M_1 \| M_3 \preceq_m M_2 \| M_4$.

As it is proven in [44], the conjunction may not exist for non-deterministic MTSs, so we assume determinism, that is, for all $p \in P$ and $a \in A$, $|\{p' \in P | p \xdashrightarrow{a} p'\}| \leq 1$. We define a pseudo MTS (pMTS) to be a MTS $M = (P, \longrightarrow, \dashrightarrow, p_0)$ with a fresh state $\perp$. The definition of conjunction is as follows. If $(p, a, p') \in \dashrightarrow \setminus \longrightarrow$, we write $p \xdashrightarrow{a}{}^{?} p'$

Definition 12. *Let $M_1 = (P_1, \longrightarrow_1, \dashrightarrow_1, p_0^1)$ and $M_2 = (P_2, \longrightarrow_2, \dashrightarrow_2, p_0^2)$ be deterministic MTSs over the same alphabet. The conjunction of M_1 and M_2 is the pMTS $M_1 \wedge M_2 = (P_1 \times P_2, \longrightarrow, \dashrightarrow, (p_0^1, p_0^2))$, that is defined as follows: for each $(p_1, p_2) \in P_1 \times P_2$,*

$$\frac{p_1 \xrightarrow{a}_1 p_1' \quad p_2 \overset{a}{\dashrightarrow}_2 p_2'}{(p_1,p_2) \xrightarrow{a} (p_1',p_2')} \qquad \frac{p_1 \overset{a}{\dashrightarrow}_1 p_1' \quad p_2 \xrightarrow{a}_2 p_2'}{(p_1,p_2) \xrightarrow{a} (p_1',p_2')} \qquad \frac{p_1 \overset{a}{\dashrightarrow}_1 p_1' \quad p_2 \overset{a}{\dashrightarrow}_2 p_2'}{(p_1,p_2) \overset{a}{\dashrightarrow} (p_1',p_2')}$$

$$\frac{p_1 \overset{a}{\dashrightarrow}_1 ? \quad p_2 \not\overset{a}{\dashrightarrow}_2}{(p_1,p_2) \not\overset{a}{\dashrightarrow}} \qquad \frac{p_1 \not\overset{a}{\dashrightarrow}_1 \quad p_2 \overset{a}{\dashrightarrow}_2 ?}{(p_1,p_2) \not\overset{a}{\dashrightarrow}} \qquad \frac{p_1 \xrightarrow{a}_1 p_1' \quad p_2 \not\overset{a}{\dashrightarrow}_2 p_2'}{(p_1,p_2) \xrightarrow{a} \bot}$$

$$\frac{p_1 \not\overset{a}{\dashrightarrow}_1 p_1' \quad p_2 \xrightarrow{a}_2 p_2'}{(p_1,p_2) \xrightarrow{a} \bot}$$

In cases where M_1 requires a transition on a and M_2 forbids it, the conjunction will introduce inconsistencies by introducing a must-transition to the state $\bot$. Example 3 shows the use of a so-called pruning operator ρ [45], that removes these inconsistencies. ρ may introduce new inconsistencies, so it should be applied until a fixpoint is reached.

Example 3. *Consider MTSs M_1 and M_2 in Fig. 8a and 8b. They both specify different coffee machines with the crucial difference that M_1 requires tea when a coin is inserted while M_2 does not allow it. Fig. 8c illustrates the conjunction.*

State p_2 of $M_1 \wedge M_2$ allows an inconsistency $p_2 \xrightarrow{tea} \bot$. Since the transition $p_1 \overset{coin}{\dashrightarrow} p_2$ is optional, we remove it. The results an be seen in Fig. 8d.

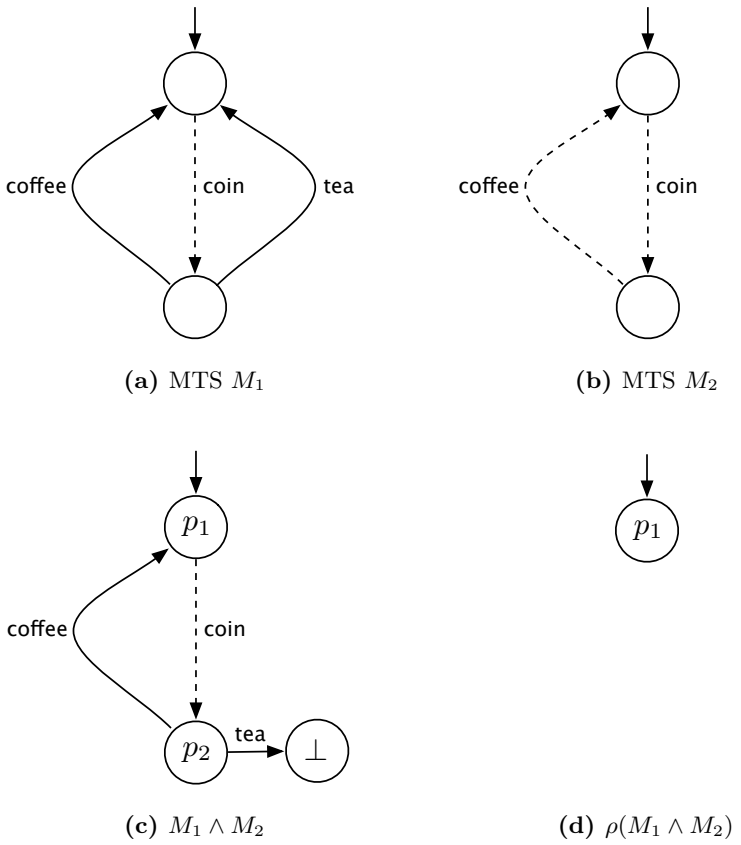
4 Probabilistic extensions of Modal Transition Systems

We now turn the attention to specification theories for probabilistic system by presenting, in this and the following sections, Interval Markov Chains, Constraint Markov Chains, and Abstract Probabilistic Automata. To describe the theory consistently and intuitively, the notations may deviate from the papers included in the thesis. This difference is purely cosmetic, and what we shortly present is equivalent to the original theory.

4.1 Interval Markov Chains

As [46] points out, e.g. LTSs does not have the expressiveness to express how frequently a transition occurs. To express the probabilistic behaviour of system components and the system as a whole, probabilistic specifications theories are needed.

A requirement that coffee is produced with a 98% chance can be modelled as in Figure 9. This is a MC as defined in Definition 1. For MCs, transitions are not labeled with actions, but instead labeled with probabilities. We will discuss the extension with actions in Section 4.3. As it is the tradition, and as can be seen in Figure 9, states are labelled with atomic propositions. This is done to be able to distinguish states from each other. In this thesis, we only consider discrete time i.e. a state change



(a) MTS M_1 **(b)** MTS M_2

(c) $M_1 \wedge M_2$ **(d)** $\rho(M_1 \wedge M_2)$

Figure 8: Conjunction followed by pruning

happens for each time step. In continuous time, the MC can reside in a state for an exponentially distributed amount of time. A Continuous-Time MC [47] is a MC equipped with a function E assigning to each state s a positive real number $E(s)$. Now the probability for a state change within the next t time units, given the present state s, is $1 - e^{-E(s)t}$, that is, the probability of moving to state s' within the next t time units is $\pi(s)(s') \left(1 - e^{-E(s)t}\right)$.

MCs are assumed to have the Markov property i.e. the probability for moving from one state to another, does not depend on the previous transitions or in other words, the MC is memoryless. This is a reasonable assumption for many systems. One could claim that the weather tomorrow relies on the weather today, but also the weather of yesterday. In such a case the Markov property is not satisfied.

Seen from the perspective of a designer, it could be the case that coffee should be produced with a 98% chance, but also that nearly 98% would suffice i.e. a probability in the interval $[0.98 - \epsilon, 0.98 + \epsilon]$ for some $\epsilon > 0$. Modeling such requirements calls for the notion of Interval Markov Chains (IMCs). In this model, the requirement of fixed transition probabilities is lifted to allow intervals of probabilities. In the end of this

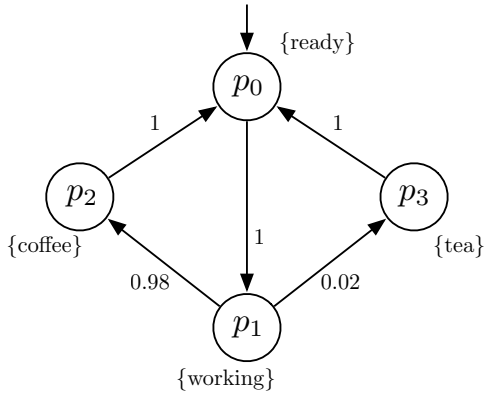


Figure 9: A Markov chain

section, the relation between IMCs and MTSs will be established.

Let $\mathsf{Intervals}_{[0,1]}$ be the set of all closed, half-open and open intervals included in $[0, 1]$.

Definition 13 (Interval Markov chain, [Paper A])**.** *An Interval Markov Chain (IMC) is a tuple $I = (Q, q_0, \varphi, A, V_I)$, where Q is a set of states containing the initial state q_0, A is a set of atomic propositions, $V_I : Q \to 2^A$ is a state valuation, and $\varphi : Q \to (Q \to \mathsf{Intervals}_{[0,1]})$, which for each $q \in Q$ and $q' \in Q$ gives an interval of probabilities.*

Given a state $q \in Q$ and a distribution $\sigma \in \mathrm{Dist}(Q)$, we say that $\sigma \in \varphi(q)$ if and only if $\sigma(q') \in \varphi(q)(q')$ for all $q' \in Q$. Notice that MC is an IMC in which all intervals are closed point intervals and all state valuations being singletons. For closed single point intervals $[p, p]$, we use the shorthand p. Figure 10 shows an IMC. The IMC specifies that the coffee machine should deliver coffee with at least a probability 0.98.

IMCs were introduced in [46] as an extension of MTSs, and have since been used as an abstraction for probabilistic systems [47, 48, 49]. The approach of [48], extended to specify Continuous-Time MCs in [47], is almost similar to our approach, but they do not investigate the compositionality of IMCs, as well as the complexity results that will be presented later in this section. In [49], Abstract Interactive Markov Chains are defined as a specification formalism for interactive Markov Chains that extends Continous-Time Markov Chains with action transitions. Parallel composition is considered, but not conjunction.

Satisfaction

For modal transition systems, modal refinement is the universal relation between MTSs and between LTSs and MTSs. For IMCs, there are separate relations for comparing specifications and for comparing implementations and specifications. The latter is the notion of satisfaction.

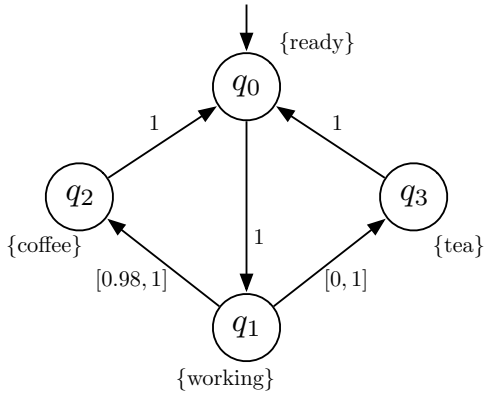


Figure 10: An Interval Markov Chain

The main ingredient of satisfaction is redistribution. Let $C = (P, p_0, \pi, A, V_C)$ be a MC and let $I = (Q, q_0, \varphi, A, V_I)$ be an IMC, and let $p \in P$ and $q \in Q$. Given a relation $\mathcal{R} \subseteq P \times Q$, we intuitively say that a distribution $\pi(p)$ can be redistributed to $\varphi(q)$, if states in P can be mapped to states in Q related by $\mathcal{R}$ and that the probability distribution induced by the mapping will conform with $\varphi(q)$.

Definition 14 ($\lhd^{\delta}_{\mathcal{R}}$)**.** *Let $C = (P, p_0, \pi, A, V_C)$ be a MC and let $I = (Q, q_0, \varphi, A, V_I)$ be an IMC, and let $p \in P$ and $q \in Q$. Given a function $\delta : P \to (Q \to [0, 1])$, and a binary relation $\mathcal{R} \subseteq P \times Q$, $\pi(p)$ is redistributed to $\varphi(q)$ with respect to $\mathcal{R}$ and δ, denoted as $\pi(p) \lhd^{\delta}_{\mathcal{R}} \varphi(q)$, if and only if*

1. *for all $p' \in P$, if $\pi(p)(p') > 0$, then $\delta(p')$ defines a distribution on Q,*
2. *for all $q' \in Q$, $\sum_{p' \in P} \pi(p)(p')\delta(p')(q') \in \varphi(q)(q')$, and*
3. *if $\delta(p')(q') > 0$, then $p' \,\mathcal{R}\, q'$.*

We write $\pi(p) \lhd_{\mathcal{R}} \varphi(q)$ if and only if there exists a function δ such that $\pi(p) \lhd^{\delta}_{\mathcal{R}} \varphi(q)$. Such δ is called a correspondence function*.*

An example of redistribution can be seen in Figure 11. The dashed ellipses show the relation given by $\mathcal{R}$. The correspondence function δ witnesses the redistribution by assigning proportions to pairs of states i.e. all probability mass assigned to state p_1 by $\pi(p')(p_1)$ is redistributed to q_2; clearly $1 \cdot \frac{1}{3} \in \left[\frac{3}{10}, \frac{2}{5}\right]$. It holds that $\pi(p') \lhd^{\delta}_{\mathcal{R}} \varphi(q')$.

Satisfaction, as defined originally in [46], is defined as follows:

Definition 15 (Satisfaction)**.** *Let $C = (P, p_0, \pi, A, V_C)$ be a MC and let $I = (Q, q_0, \varphi, A, V_I)$ be an IMC. A relation $\mathcal{R} \subseteq P \times Q$ is called a* satisfaction relation *if and only if whenever $p \,\mathcal{R}\, q$ then*

- *their valuation sets agree: $V_C(p) = V_I(q)$, and*

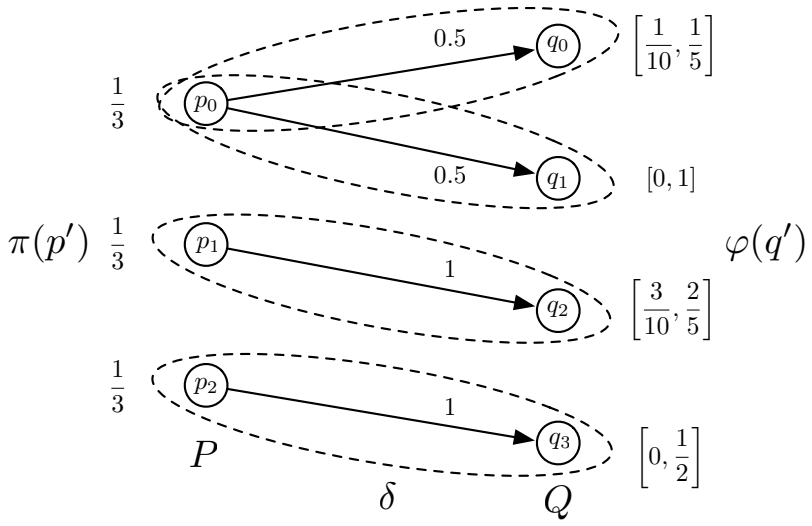


Figure 11: Redistribution

- *there exists a correspondence function* $\delta : P \to (Q \to [0,1])$ *such that* $\pi(p) \lhd_{\mathcal{R}}^{\delta} \varphi(q)$.

We say that C *satisfies* I*, written* $C \models I$*, if and only if there exists a satisfaction relation containing* (p_0, q_0)*, and call* C *an implementation of* I*. We adopt the notation of MTSs, and denote the set of implementations of* I *by* $[\![I]\!]$.

Clearly, the MC in Figure 9 satisfies the IMC in Figure 10 as $0.98 \in [0.98, 1]$; a valid satisfaction relation is $\{(p_0, q_0), (p_1, q_1), (p_2, q_2), (p_2, q_2)\}$.

Notions of refinement

The standard notion of refinement, called strong refinement, is defined along the lines of satisfaction. For a pair (p, q) in a satisfaction relation, there is exactly one distribution $\pi(p)$ that has to be redistributed. For a pair (q, s) in a strong refinement, we require that there exists a single correspondence function such that all distributions $\sigma \in \varphi_1(q)$ can be redistributed to $\varphi_2(s)$.

Definition 16 (Strong Refinement)**.** *Let* $I_1 = (Q, q_0, \varphi_1, A, V_1)$ *and* $I_2 = (S, s_0, \varphi_2, A, V_2)$ *be two IMCs. A relation* $\mathcal{R} \subseteq Q \times S$ *is called a* strong refinement relation *if and only if whenever* $q \mathcal{R} s$*, then*

- *their valuation sets agree:* $V_1(q) = V_2(s)$*, and*
- *there exists a correspondence function* $\delta : Q \to (S \to [0,1])$ *such that for all* $\sigma \in \varphi_1(q)$*, it holds that* $\sigma \lhd_{\mathcal{R}}^{\delta} \varphi_2(s)$.

I_1 strongly refines I_2, written $I_1 \preceq_S I_2$, if and only if there exists a strong refinement relation containing (q_0, s_0).

In [Paper A], we also present two weaker notions of refinement:

1. Weak refinement $\preceq$: instead of a single correspondence function such that all distributions $\sigma \in \varphi_1(q)$ can be redistributed to $\varphi_2(s)$, we allow one correspondence function to be chosen for each distribution $\sigma \in \varphi_1(q)$. Strong refinement implies weak refinement.

2. Thorough refinement: we say that IMC I_1 thoroughly refines I_2 if and only if $[\![I_1]\!] \subseteq [\![I_2]\!]$.

Both weak and strong refinement implies thorough refinement, but as for MTS, they are sound, but not complete with respect to thorough refinement. Analogous to MTS, completeness is obtained for a special class of IMCs, that we call deterministic IMCs with the following definition: An IMC $I = (Q, q_0, \varphi, A, V)$ is deterministic if and only if for all states $q, r, s \in Q$, if there exist a probability distribution $\sigma \in \varphi(q)$ such that $\sigma(r) > 0$ and a probability distribution $\rho \in \varphi(q)$ such that $\rho(s) > 0$, then $V(r) \neq V(s)$. For deterministic IMCs, it is known that the three notions of refinement coincide. In [Paper A], the problem of checking thorough refinement was shown to be EXPTIME-complete.

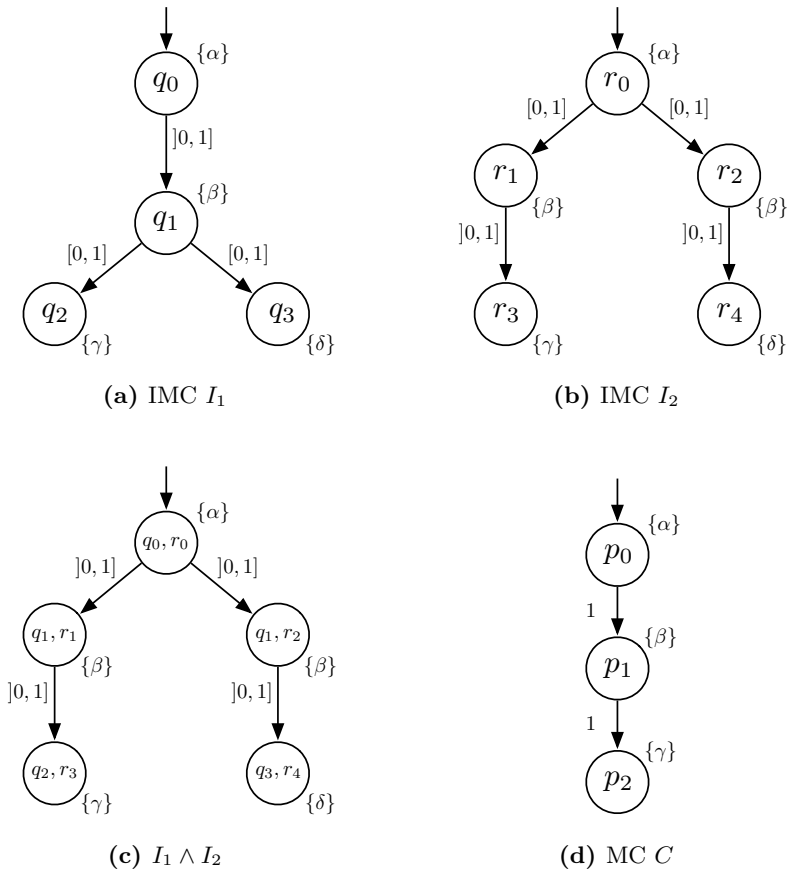
Non-closure problems

Given IMCs M_1 and M_2, the conjunction operator $\wedge$ should satisfy that $M_1 \wedge M_2 \preceq M_1$, $M_1 \wedge M_2 \preceq M_2$, and for all IMCs M such that $M \preceq M_1$ and $M \preceq M_2$, then $M \preceq M_1 \wedge M_2$. This states that conjunction is the greatest lower bound with respect to refinement, and implies that $[\![M_1 \wedge M_2]\!] = [\![M_1]\!] \cap [\![M_2]\!]$. When considering IMCs, a naive approach is to conjoin intervals by taking their intersection. We show that this approach does not work.

Example 4. *Consider IMCs I_1 and I_2 in Fig. 12a and 12b, and their conjunction by interval intersection in Fig. 12c. The MC C (Fig. 12d) satisfies both I_1 and I_2 and should therefore also satisfy $I_1 \wedge I_2$. However, this is not the case, and hence, conjunction by interval intersection does not satisfy the requirement for a conjunction operator.*

Similar examples in [Paper A] and [Paper B] show that IMCs are not closed under conjunction and parallel composition. In [Paper C], it was shown that there are IMCs M_1 and M_2 for which there exists no IMC I with $[\![I]\!] = [\![M_1]\!] \cap [\![M_2]\!]$. This is a strong result as it abstracts away from the specific definition of the conjunction operator, and says that no conjunction operator will give rise to an IMC in this example.

The closure problem has lead us to study the common implementation problem, which is similar to asking whether there exists a MC satisfying the conjunction:



(a) IMC I_1 **(b)** IMC I_2

(c) $I_1 \wedge I_2$ **(d)** MC C

Figure 12: Conjunction of IMCs

Problem (Common Implementation (CI)). *Given $k > 1$ IMCs I_i, $i = 1, \ldots, k$, does there exist a MC C such that $C \models I_i$ for all i?*

Answering the question involves constructing a relation with properties that bear a resemblance to the above definition of conjunction, but it is not necessary to construct the actual conjunction. In [Paper A], we have shown that the CI problem is EXPTIME-complete for unbounded k, and in PTIME for a constant k. This latter results carries over to consistency checking, i.e. checking whether an IMC C has at least one implementation can be formulated as the CI problem for $k = 2$ IMCs, namely C and itself.

Relation to MTSs

Building upon an idea of [46], we present a translation in [Paper A] that transforms a MTS M into an IMC $\widehat{M}$ with the property that, given a LTS L, it holds that $L \preceq_m M \Leftrightarrow [\![\widehat{L}]\!] \subseteq [\![\widehat{M}]\!]$. Intuitively, a may-transition is modelled as a probabilistic

transition with the interval $[0, 1]$, whereas a must-transition is modeled as the interval $]0, 1]$. This transformation is of great importance, since it is used for proving lower complexity bounds for checking thorough refinement and the common implementation problem by reductions from the same problems for MTSs [43, 50].

4.2 Constraint Markov Chains

A Constraint Markov Chain (CMCs) is an extension of an IMC that overcomes the non-closure problems by generalizing intervals to general constraints. Also the concept of state labeling is generalized by allowing a set of possible sets of atomic propositions for an implementation to satisfy.

Formally, a Constraint Markov Chain is defined as follows:

Definition 17 (Constraint Markov Chain)**.** *A Constraint Markov Chain is a tuple* $S = (Q, q_0, \varphi, A, V)$*, where* Q *is a set of states containing the initial state* q_0*,* A *is a finite set of atomic propositions,* $V : Q \to 2^{2^A}$ *is a set of admissible state valuations and* $\varphi : Q \to \text{Dist}(Q) \to \{0, 1\}$ *is a* constraint function*.*

A constraint $\varphi(q)$ is simply an indicator function over $\text{Dist}(Q)$. Given $\mu \in \text{Dist}(Q)$, $\varphi(q)(\mu) \in \{0, 1\}$ specifies whether μ is allowed (1) or disallowed (0). We choose to write constraints as predicates. Consider a CMC on state space $Q = \{q_1, q_2, q_3\}$ and let $q \in Q$. As an example, a constraint

$$\varphi(q)(x_{q_1}, x_{q_2}, x_{q_3}) \equiv (0.1 \leq x_{q_1} \leq 0.2 \vee 0.7 \leq x_{q_1} \leq 0.9) \wedge \sum_{i=1}^{3} x_{q_i} = 1 \tag{3}$$

translates to the set $\{\mu \in \text{Dist}(Q) | 0.1 \leq \mu(q_1) \leq 0.2 \vee 0.7 \leq \mu(q_1) \leq 0.9\}$. Figure 13 illustrates the constraint.

In [51], Parametric Probabilistic Transition Systems are presented. In this model, transitions are labeled with multivariate polynomials and the relation to MC is obtained by assigning the variables of the polynomials to induce a distribution. The authors consider the problem of finding variable assignments that satisfy certain properties, and do not consider compositionality or notions of refinement.

Consider again the example from Section 4.1, that a designer of a coffee machine specifies that coffee should produced with a 98% chance, but that nearly 98% also will suffice, i.e. a probability in the interval $[0.98 - \epsilon, 0.98 + \epsilon]$. The motivation for generalizing intervals to general constraints is not as clear as generalizing fixed probabilities to intervals. Indeed, an example of a designer specifying that coffee should be produced with a probability 0.98 or 0.6 seems farfetched. The main motivation for introducing CMCs lies in the closure properties.

Given a state q in a CMC, the valuation $V(q)$ is a set of sets i.e. $V(q) = \{S_1, \ldots, S_k\}$ for $k \geq 1$ or $V(q) = \emptyset$. For a state p in a MC C to implement q, it must take a valuation in $V(q)$ i.e. $V_C(p) \in V(q)$. Notice that $V(q) = \emptyset$ can not be implemented. This approach is more expressive than the labeling function of e.g. [47, 48]. Here they consider a function $V_m : Q \times A \to \mathbb{B}_3$, with $\mathbb{B}_3 = \{\bot, ?, \top\}$ being a complete lattice

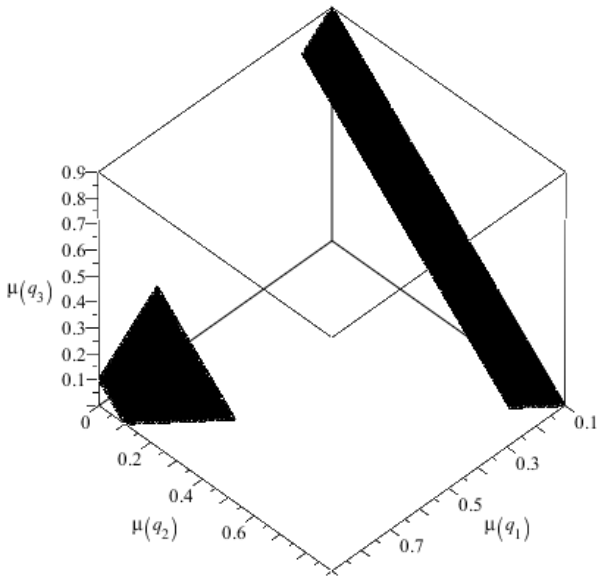


Figure 13: Illustrating a constraint

with the ordering $\bot < ? < \top$. This function deems atomic propositions to exist (must, $\top$), may exist (may, ?), or not exist (must not, $\bot$) in the valuation of an implementing state of a MC.

In the state valuation $V(q) = \{\{a\}, \{b\}\}$, it is expressed that a or b should exclusively hold. V_m can deem a and b as may-propositions, but this could result in that a holds, b holds, a and b holds or a and b does not hold, where the last two options are unwanted. Given a state q in a CMC, consider the set

$$R_{V_m}(q) = \left\{B \in 2^A | \{a \in A | V_m(q,a) = \top\} \subseteq B \subseteq \{a \in A | V_m(q,a) \geq ?\}\right\}.$$

This is the set of possible realizations of V_m in the state q. As an example, consider $A = \{a, b, c\}$, a state q and V_m defined as $\{V_m(q,a) = \top, V_m(q,b) = ?, V_m(q,c) = \bot\}$; here $R_{V_m}(q) = \{\{a\}, \{a,b\}\}$. As shown for transitions in modal transition systems in [41], $R_{V_m}(q)$ is closed under intersection, union and convexity i.e. if $X, Z \in R_{V_m}(q)$ and there exists a set $Y \in 2^A$ such that $X \subseteq Y \subseteq Z$, then $Z \in R_{V_m}(q)$. The state valuations we consider in our approach, need not to be closed under these operations.

Like IMCs, CMCs feature three notions of refinement that are defined analogously.

Parallel composition and conjunction

Conjunction and parallel composition for CMCs are defined as follows.

- Conjunction: Let $S_1 = (Q_1, q_0, \varphi_1, A, V_1)$ and $S_2 = (Q_2, s_0, \varphi_2, A, V_2)$ be CMCs and let $q \in Q_1$ and $s \in Q_2$. The conjunction of S_1 and S_2 is the CMC $S_1 \wedge S_2 = (Q_1 \times Q_2, (q_0, s_0), \varphi_\wedge, A, V)$, that is defined as follows:

– We construct $\varphi_\wedge(q,s)$ of the product state (q,s) as

$$\varphi_\wedge(q,s)(\sigma_\wedge) \equiv \varphi_1(q)\left(q' \mapsto \sum_{s' \in S} \sigma_\wedge(q',s')\right) \wedge \varphi_2(s)\left(s' \mapsto \sum_{q' \in Q} \sigma_\wedge(q',s')\right).$$

– We define $V(q,s) = V_1(q) \cap V_2(s)$.

- Parallel composition: Let $S_1 = (Q_1, q_0, \varphi_1, A_1, V_1)$ and $S_2 = (Q_2, s_0, \varphi_2, A_2, V_2)$ be CMCs with $A_1 \cap A_2 = \emptyset$ and let $q \in Q_1$ and $s \in Q_2$. The parallel composition of S_1 and S_2 is the CMC $S_1 \| S_2 = (Q_1 \times Q_2, (q_0, s_0), \varphi_\|, A, V)$, that is defined as follows:

  – We construct $\varphi_\|(q,s)$ of the product state (q,s) as

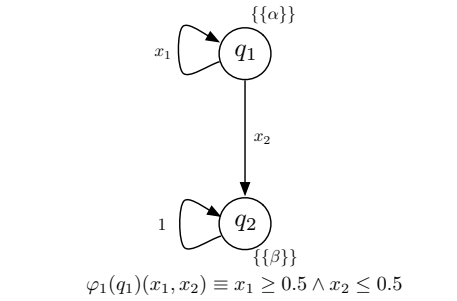
$$\varphi_\|(q,s)(\sigma_\|) \equiv \exists \sigma_1 \in \mathrm{Dist}(Q_1) : \varphi_1(q)(\sigma_1), \exists \sigma_2 \in \mathrm{Dist}(Q_2) : \varphi_2(s)(\sigma_2), \forall q' \in Q, s' \in S : \sigma_\|(q',s') = \sigma_1(q')\sigma_2(s').$$

  – We define $V(q,s) = V_1(q) \cup V_2(s)$.

The conjunction treats probabilities by allowing $\sigma_\wedge$ if its marginal distributions are allowed in S_1 and S_2, respectively. Parallel composition treats probabilities by allowing $\sigma_\|$, if it can be computed as the product of distributions from S_1 and S_2.

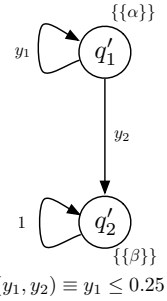
Conjunction satisfies that $[\![S_1 \wedge S_2]\!] = [\![S_1]\!] \cap [\![S_2]\!]$. Parallel composition satisfies that given CMCs S_1, S_2, S_3, and S_4, for which it holds that $S_1 \preceq S_3$ and $S_2 \preceq S_4$, we have $S_1 \| S_2 \preceq S_3 \| S_4$. The conjunction can introduce inconsistencies. Calculating $V_1(q) \cap V_2(s)$ may yield the empty set, which means that (q,s) can not be implemented. These so-called valuation inconsistencies must be removed, and that could in turns create constraint-inconsistencies, that is, constraints that can not be satisfied. To this end, we have in [Paper B] defined a pruning operator β that can remove inconsistencies. Like pruning for MTSs, the pruning operator may introduce new inconsistencies, so it should be applied until a fixpoint is reached. It holds for all CMCs S, that $[\![S]\!] = [\![\beta^*(S)]\!]$.

Example 5. *Consider the CMCs in Figure 14a and 14b and their conjunction in Figure 14c. Requirements for variables to sum to* 1 *have been left out to avoid clutter. States* (q_1, q_2) *and* (q_2, q_1) *have empty state valuations and are therefore inconsistent. The constraint of state* (q_1, q_1') *in* $\beta(S_1 \wedge S_2)$ *(see Figure 14d) asks for the existence of a distribution satisfying* $\varphi'_\wedge(q_1, q_1')$*, that gives zero probability to states* (q_1, q_2) *and* (q_2, q_1)*. Such a distribution does not exist, since it requires that* $z_{1,1'} \geq 0.5$ *and* $z_{1,1'} \leq 0.25$*, simultaneously. Therefore,* (q_1, q_1') *in* $\beta(S_1 \wedge S_2)$ *is inconsistent, and applying* β *again, reaching the fixpoint, yields an empty CMC.*



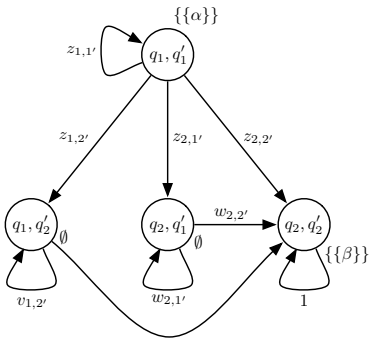
$\varphi_1(q_1)(x_1, x_2) \equiv x_1 \geq 0.5 \wedge x_2 \leq 0.5$

(a) CMC S_1



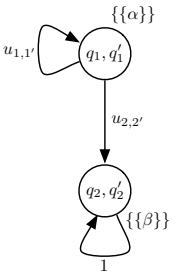
$\varphi_2(q'_1)(y_1, y_2) \equiv y_1 \leq 0.25 \wedge y_2 \geq 0.75$

(b) CMC S_2



$\varphi_\wedge(q_1, q'_1)(z_{1,1'}, z_{1,2'}, z_{2,1'}, z_{2,2'}) \equiv z_{1,1'} + z_{1,2'} \geq 0.5 \wedge$
$z_{2,1'} + z_{2,2'} \leq 0.5 \wedge z_{1,1'} + z_{2,1'} \leq 0.25 \wedge$
$z_{1,2'} + z_{2,2'} \geq 0.75$
$\varphi_\wedge(q_1, q'_2)(v_{1,1'}, v_{1,2'}, v_{2,1'}, v_{2,2'}) \equiv v_{1,1'} + v_{1,2'} \geq 0.5 \wedge$
$v_{2,1'} + v_{2,2'} \leq 0.5 \wedge v_{1,1'} + v_{2,1'} = 0 \wedge$
$v_{1,2'} + v_{2,2'} = 1$
$\varphi_\wedge(q_2, q'_1)(w_{1,1'}, w_{1,2'}, w_{2,1'}, w_{2,2'}) \equiv w_{1,1'} + w_{1,2'} = 0 \wedge$
$w_{2,1'} + w_{2,2'} = 1 \wedge w_{1,1'} + w_{2,1'} \leq 0.25 \wedge$
$w_{2,1'} + w_{2,2'} \geq 0.75$

(c) $S_1 \wedge S_2$



$\varphi'_\wedge(q_1, q'_1)(u_{1,1'}, u_{2,2'}) \equiv \exists z_{1,1'}, z_{1,2'}, z_{2,1'}, z_{2,2'} :$
$z_{1,2'} = z_{2,1'} = 0 \wedge$
$z_{1,1'} = u_{1,1'} \wedge z_{2,2'} = u_{2,2'} \wedge$
$z_{1,1'} + z_{1,2'} \geq 0.5 \wedge z_{2,1'} + z_{2,2'} \leq 0.5 \wedge$
$z_{1,1'} + z_{2,1'} \leq 0.25 \wedge z_{1,2'} + z_{2,2'} \geq 0.75$

(d) $\beta(S_1 \wedge S_2)$

Figure 14: Conjunction followed by pruning

Closure properties

As shown in Section 4.1, interval constraints do not suffice for closure under conjunction and parallel composition. Considering general constraints gives closure for CMCs under parallel composition and conjunction, but in [Paper B] we have studied the problem of determining which types of constraints that ensure closure.

As it turns out, linear constraints suffice for closure under conjunction. Not surprisingly, as parallel composition utilizes multiplication of variables, structurally composing linear constraints gives rise to polynomial constraints. That is, polynomial constraints are closed under parallel composition, but linear constraints are not.

Notions of abstraction

The CMC formalism features two notions of abstractions. Given a CMC S the intuition of abstraction is to obtain a CMC S' such that $S \preceq S'$. This CMC S' may e.g. be especially efficient in terms of computations or may hide away non-essential details. Notice that abstraction serves the dual purpose of refinement.

- Constraint-based abstraction: The constraints of a CMC may be very complex, and to efficiently reason about them, one can think of the smallest interval-abstraction that contains a constraint.

  Definition 18 (Constraint-based abstraction)**.** *Let $S = (Q, q_0, \varphi, A, V)$ be a CMC. The constraint-abstracted CMC $\chi(S) = (Q, q_0, \varphi', A, V)$ is defined such that*

  $$\varphi'(q)(\mu) \equiv \bigwedge_{q' \in Q} \mu(q') \in I^q_{q'} \wedge \sum_{q' \in Q} \mu(q') = 1,$$

  where, for all $q' \in Q$, $I^q_{q'}$ are the smallest closed intervals such that $\forall \mu' \in \mathrm{Dist}(Q), \varphi(q)(\mu') \Rightarrow \bigwedge_{q' \in Q} \mu'(q') \in I^q_{q'}$.

  Notice that, but for the valuations, the obtained CMC is in fact an IMC, and it is also clear that $S \preceq \chi(S)$.

- State-based abstraction: Composition of CMCs suffers from the state space explosion problem. To address this problem we have defined an abstraction operator that collapses states in the concrete state space Q to form the abstract state space Q'. A surjection $\alpha : Q \to Q'$ with the property that $Q = \bigcup_{q' \in Q'} \alpha^{-1}(q')$ is called an abstraction function. We define the abstraction of a distribution μ as $\alpha(\mu)(q') = \sum_{q \in \alpha^{-1}(q')} \mu(q)$ for all $q' \in Q'$.

  Definition 19 (State-based abstraction)**.** *Let $S = (Q, q_0, \varphi, A, V)$ be a CMC and let $\alpha : Q \to Q'$ be a state abstraction function. The CMC $\alpha(S) = (Q, \alpha(q_0), \varphi', A, V')$ is induced by α as follows:*

  $$\varphi'(q')(\mu') \equiv \exists \mu \in \mathrm{Dist}(Q) : \mu' = \alpha(\mu) \wedge \bigvee_{q \in \alpha^{-1}(q')} \varphi(q)(\mu), \text{ and}$$
  $$V'(q') = \bigcup_{q \in \alpha^{-1}(q')} V(q).$$

  Again, it is clear that the abstraction works as intended i.e. $S \preceq \alpha(S)$.

Example 6. *The constraint-abstraction of $\varphi(q)$ in Equation (3) is*

$$\varphi'(q)(x_{q_1}, x_{q_2}, x_{q_3}) \equiv x_{q_1} \in [0.1, 0.9] \wedge x_{q_2} \in [0, 0.9] \wedge x_{q_3} \in [0, 0.9] \wedge \sum_{i=1}^{3} x_{q_i} = 1.$$

Figure 15 illustrates the abstracted constraint, obtained by "filling out" the polygon.

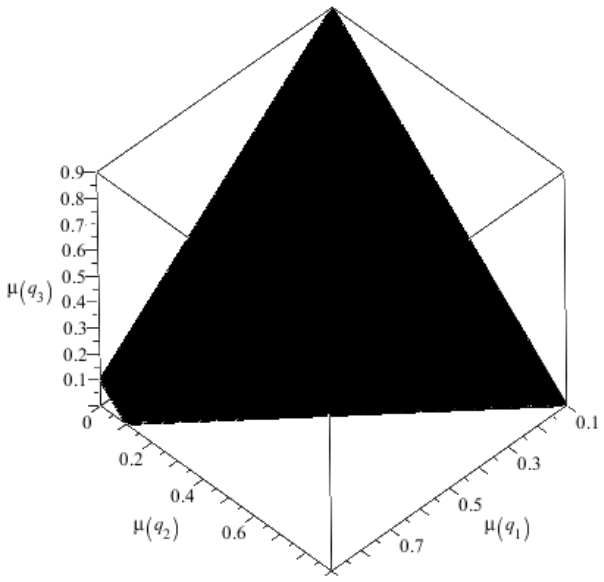


Figure 15: Abstracting a constraint

In [Paper C], we have shown that the parallel composition of two abstractions, $\alpha_1(S_1)\|\alpha_2(S_2)$, is equivalent to abstracting $S_1\|S_2$ with the cartesian product of abstraction functions $\alpha_1 \times \alpha_2$. This property, phrased as "local abstraction = global abstraction", shows the usefulness of state-based abstraction. A similar result holds for constraint-based abstraction, but as $\chi(S_1)\|\chi(S_2)$ is not in general an IMC, we only obtain that $\chi(S_1)\|\chi(S_2) \preceq \chi(S_1\|S_2)$.

4.3 Abstract Probabilistic Automata

A Probabilistic Automaton (PA) is a mixture of a LTS and a MC in the sense that the formalism encompasses both action transitions and next-state distributions. PAs are a widely recognized mathematical framework for the specification and analysis of systems with non-deterministic and stochastic behavior developed by Segala and Lynch [52]. We use the variant called a *simple* PA [53], and add the additional feature that states are labelled with atomic propositions.

PAs are very similar to Markov Decision Processes [54], but supports both internal and external non-determinism [55]. In [6, 53], PAs has been extended to Probabilistic I/O Automata (PIOA), and have been used for verification of protocols [4] and for analysing randomized distributed algorithms [56].

Figure 16 illustrates the running example for PAs with respect to implementations. Notice that a MC is a PA on one action, say α, where, for each $s \in S$, there exists exactly one triple (s, α, μ) such that $L(s, a, \mu) = \top$.

When composing PAs (and later APAs), we want to adopt a parallel composition

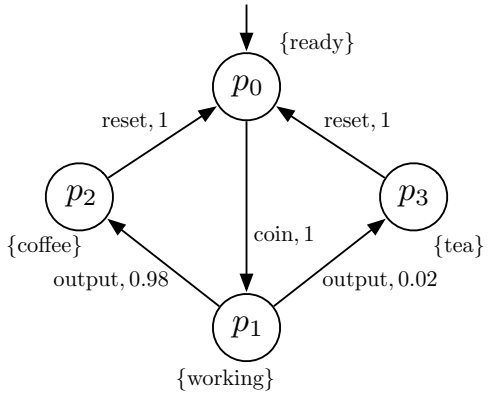


Figure 16: A probabilistic automaton

operator that allows two PAs to synchronize on a set of actions, while interleaving for the remaining actions. This synchronization paradigm, called handshaking [15, 57], is important for concurrent systems that are not fully synchronous. Figure 17 shows the desired result of the parallel composition of two PAs. The empty synchronization is used, so the result is entirely interleaving. The parallel composition $N_1 \|_\emptyset N_2$ uses non-determinism on actions to represent interleaving. As one can consider a CMC as a PA with only one action α, the parallel composition of CMCs correspond to their parallel composition as PAs with $\|_{\{\alpha\}}$, and it is clear that CMC conjunction is fully synchronous, meaning that interleaving is not expressible, due to the lack of non-determinism on actions.

Extending PA with state labelings in $2^{2^{AP}}$ and general probability constraints follows naturally. Additionally, action transitions are labeled with modalities similarly to MTSs.

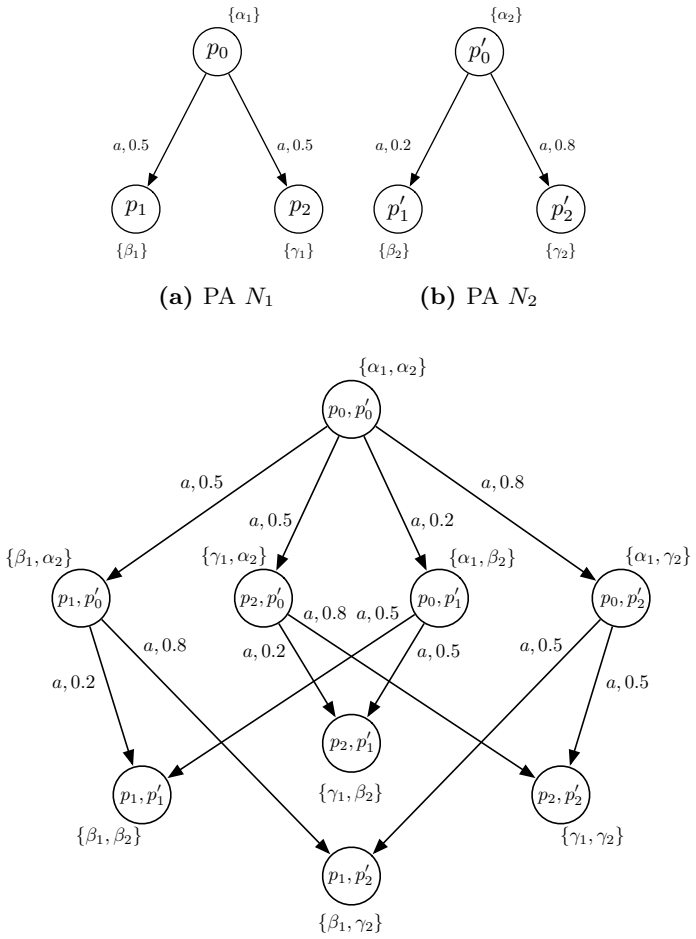
Definition 20. *An Abstract Probabilistic Automaton is a tuple $N = (S, A, L, AP, V, s_0)$, where S is the set of states containing the initial state s_0, A is a finite set of actions, $L : S \times A \times C(S) \longrightarrow \mathbb{B}_3$ is a three-valued state-constraint function with $\mathbb{B}_3 = \{\bot, ?, \top\}$ being a complete lattice with the ordering $\bot < ? < \top$, AP is a finite set of atomic propositions, and V: $S \to 2^{2^{AP}}$ is a state-labeling function.*

As the definition specifies, a transition on an action a and a constraint φ is deemed either as a must transition ($\top$), a may transition ($?$), or as not allowed ($\bot$). We denote by $Sat(\varphi)$ the set of distributions satisfying φ.

Figure 18 illustrates an example of an APA. The constraint φ_x specifies that the probability for getting coffee is at least 0.98.

APAs feature determinism in two kinds. An APA $N = (S, A, L, AP, V, s_0)$ is action-deterministic if

$$\forall s \in S, \forall a \in A : \ |\{\varphi \in C(S) | L(s, a, \varphi) \neq \bot\}| \leq 1,$$



(a) PA N_1 **(b)** PA N_2

(c) $N_1 \|_\emptyset N_2$

Figure 17: Parallel composition of PAs

and valuation-deterministic if $\forall s \in S, \forall a \in A, \forall \varphi \in C(S)$ with $L(s, a, \varphi) \neq \bot$:

$$\forall \mu', \mu'' \in Sat(\varphi), s', s'' \in S,\ \left(\mu'(s') > 0 \wedge \mu''(s'') > 0\ \Rightarrow V(s') \cap V(s'') = \emptyset\right).$$

We say that N is deterministic if it is both action-deterministic and valuation-deterministic.

The notion of APA is equipped with operations known for IMCs and CMCs such as the two presented notions of abstraction extended with handling of the modalities, parallel composition, conjunction etc.

Refinement

To see the interplay of probabilities and modalities, the notion of weak refinement is presented. First the notion of redistribution is defined for the notation of APAs. The

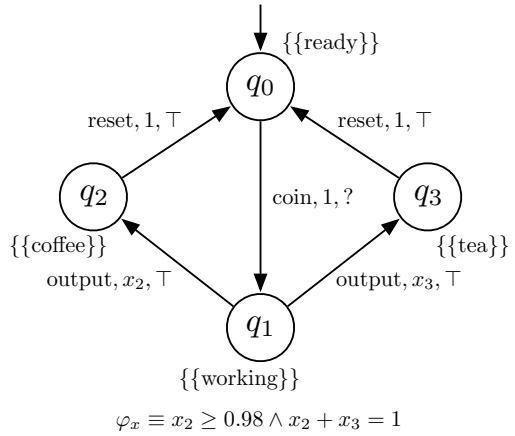


Figure 18: An abstract probabilistic automaton

definition is very similar to that of IMCs.

Definition 21. *Let S and S' be non-empty sets of states. Given $\mu \in \text{Dist}(S)$, $\mu' \in \text{Dist}(S')$, a function $\delta : S \to (S' \to [0,1])$, and a binary relation $R \subseteq S \times S'$, μ is redistributed to μ' with respect to R and δ, denoted as $\mu \Subset_R^\delta \mu'$, if and only if*

1. *for all $s \in S$, if $\mu(s) > 0$, then $\delta(s)$ is a distribution on S',*
2. *for all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$, and*
3. *if $\delta(s)(s') > 0$, then $(s, s') \in R$.*

We write $\mu \Subset_R \mu'$ if and only if there exists a function δ such that $\mu \Subset_R^\delta \mu'$. Such δ is called a correspondence function*.*

Having modal refinement in mind, we know that transitions must be matched in the modal way, and that resulting states must be related. But for APAs, resulting states are obtained by probabilities. The following definition unifies the notion of refinement for CMCs and MTSs.

Definition 22. *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP, V', s_0')$ be APAs. $R \subseteq S \times S'$ is a weak refinement relation if and only if, for all $(s, s') \in R$, the following conditions hold:*

1. $\forall a \in A. \forall \varphi' \in C(S'). L'(s', a, \varphi') = \top \implies \exists \varphi \in C(S). L(s, a, \varphi) = \top$ *and* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset_R \mu'$,
2. $\forall a \in A. \forall \varphi \in C(S). L(s, a, \varphi) \neq \bot \implies \exists \varphi' \in C(S'). L'(s', a, \varphi') \neq \bot$ *and* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset_R \mu'$, *and*

3. $V(s) \subseteq V'(s')$.

We say that N weakly refines N', denoted $N \preceq N'$, if and only if there exists a weak refinement relation relating s_0 and s'_0.

Conjunction

As mentioned earlier, the conjunction of non-deterministic MTSs may not exist [44]. For APAs, it is possible to conjoin non-deterministic APAs and obtain the greatest lower bound with respect to the so-called weak weak refinement.

Definition 23. *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP, V', s'_0)$ be APAs. $R \subseteq S \times S'$ is a weak weak refinement relation if and only if, for all $(s, s') \in R$, the following conditions hold:*

1. $\forall a \in A. \forall \varphi' \in C(S'). L'(s', a, \varphi') = \top \implies \exists \varphi \in C(S). L(s, a, \varphi) = \top$ *and* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset_R \mu'$,
2. $\forall a \in A. \forall \varphi \in C(S). L(s, a, \varphi) \neq \bot \implies \forall \mu \in Sat(\varphi). \exists \varphi' \in C(S'). L'(s', a, \varphi') \neq \bot$ *and* $\exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset_R \mu'$, *and*
3. $V(s) \subseteq V'(s')$.

We say that N weakly weakly refines N', denoted $N \preceq_W N'$, if and only if there exists a weak weak refinement relation relating s_0 and s'_0.

Only item 2 changes in comparison to the definition of weak refinement. Given a constraint φ, in weak refinement, we have to find a constraint φ' such that all distributions in φ are redistributed to distributions in φ'. However, weak weak refinement allows to choose a different φ' for each $\mu \in Sat(\varphi)$.

4.4 The tool APAC

APAC (short for APA Checker) provides tool support for the operations defined for CMCs and APAs. The functionality is built upon the SMT solver Z3 from Microsoft Research, that is able to perform quantifier elimination (QE) for linear arithmetic over the reals.

The limitation to linear arithmetic as well as the heavy complexity of performing QE are the major drawbacks. We will comment on the issues in the end of this section.

Example of use

Consider the IMC I in Figure 10 and a variant I' in Figure 19. The two IMCs specify different specifications for a coffee machine and we are interested in checking whether the specifications can be satisfied simultaneously.

As a first step we specify the IMCs in the notation of APAC. Figure 20 shows the input file. The last line symbolizes that we want APAC to decide whether the

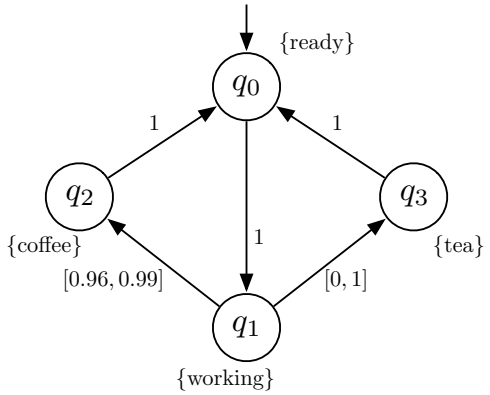


Figure 19: An Interval Markov Chain I'

```
INPUT:
Name: I;
AP:(ready,working,coffee,tea);
state 1:((ready)): x[2]=1.0;
state 2:((working)): x[1]=0.0 && x[2] = 0.0 && x[3]>=98/100;
state 3:((coffee)): x[1]=1.0;
state 4:((tea)): x[1]=1.0;

Name: Iprime;
AP:(ready,working,coffee,tea);
state 1:((ready)): x[2]=1.0;
state 2:((working)): x[1]=0.0 && x[2] = 0.0 && x[3]>=96/100 && x[3]<=99/100;
state 3:((coffee)): x[1]=1.0;
state 4:((tea)): x[1]=1.0;

check: C(I and Iprime);
```

Figure 20: Input file

conjunction of I and I' is consistent (C) i.e. whether there exists a MC C such that $C \models I \wedge I'$.

The file is passed to the tool and the output is given in Figure 21. Indeed, the conjunction is consistent.

Encoding in Z3

The tool ultimately relies on the encoding in problems solvable by Z3. As an example, consider checking weak refinement between two CMCs. Let $S_1 = (Q_1, q_0^1, \varphi_1, A, V_1)$ and $S_2 = (Q_2, q_0^2, \varphi_2, A, V_2)$ be CMCs. Assume that $Q_2 = \{q_1, \ldots, q_{|Q_2|}\}$. Let $u \in Q_1$ and $v \in Q_2$. For (u, v) to be member of a weak refinement relation we first check that $V_1(u) \subseteq V_2(v)$. Next the following formula is passed to Z3. As all variables are quantified the formula will reduce to true or false.

```
==========RESULT SUMMARY==========
I_and_Iprime is consistent.
Time elapsed: 146 ms

==================================
Running time: 376 ms
```

Figure 21: Output

$$\forall x \in \mathrm{Dist}(Q_1) : \varphi_1(u)(x) \Rightarrow \exists \delta : Q_1 \to (Q_2 \to [0,1]) : \tag{4}$$

$$\varphi_2(v)\left(\sum_{u' \in Q_2} \delta(u')(q_1), \ldots, \sum_{u' \in Q_2} \delta(u')(q_{|Q_2|})\right) \quad \wedge$$

$$\forall u' \in Q_1 : x(u) = \sum_{v' \in Q_2} \delta(u')(v') \wedge$$

$$\forall u' \in Q_1, \forall v' \in Q_2 : \delta(u')(v') \neq 0 \Rightarrow u' \,\mathcal{R}\, v'. \tag{5}$$

A special multiplication-free version of weak refinement is considered, as the original definition contains multiplication of variables. The special version of refinement is shown in [Paper C].

The above example shows the use of Z3 when all variables are quantified. In the case of constraint abstraction, certain variables are kept free. Let $u \in Q_1$.

$$\begin{aligned}
&\forall v \in Q_1 : (0 \leq L_v \leq U_v \leq 1) \wedge \\
&(\forall x \in \mathrm{Dist}(Q_1) : \varphi(u)(x) \Rightarrow \forall v \in Q_1 : L_v \leq x(v) \leq U_v) \wedge \\
&\forall (L'_1, U'_1, \ldots, L'_k, U'_k) \in [0,1]^{|Q_1|} : \\
&\qquad \Big[\big((\forall x \in \mathrm{Dist}(Q_1) : \varphi(u)(x) \Rightarrow \forall v \in Q_1 : L'_v \leq x(v) \leq U'_v) \wedge \\
&\qquad\qquad (\forall v \in Q_1 : 0 \leq L'_v \leq U'_v \leq 1) \big) \Rightarrow \\
&\qquad \big(\forall v \in Q_1 : L'_v \leq L_v \wedge U_v \leq U'_v \big) \Big].
\end{aligned}$$

Notice, that the L_u's and U_u's are not quantified, and are thus free. The formula specifies that the L_u's and U_u's should contain all admissible distributions, and that all other intervals, containing all admissible distributions, should be bigger.

Linear arithmetic

The restriction to linear arithmetic prevents us from implementing the notions of parallel composition and strong refinement for both CMCs and APAs as these notions rely inherently on multiplication of variables. To still be able to reason about the parallel

composition, we have defined the so-called linear parallel composition, that acts like parallel composition with respect to modalities and valuations, but linearly abstracts constraints.

There exists tools for non-linear QE (Redlog [58], QEPCAD [59]) but, quoting [60], it is problematic that the process of non-linear QE in practice "does not scale to systems of more than five variables". Since checking refinement on a small example of two CMCs on 5 states each requires 30 quantified variables, non-linear QE is not feasible, at the moment.

5 Thesis summary

Paper A – *Consistency and Refinement for Interval Markov Chains*

Interval Markov Chains (IMC), or Markov Chains with probability intervals in the transition matrix, are the base of a classic specification theory for probabilistic systems (Larsen and Jonsson, 1991). The standard semantics of IMCs assigns to a specification the set of all Markov Chains that satisfy its interval constraints. The theory then provides operators for deciding emptiness of conjunction and refinement (entailment) for such specifications.

In this paper we study complexity of several problems for IMCs, that stem from compositional modeling methodologies. In particular, we close the complexity gap for thorough refinement of two IMCs and for deciding the existence of a common implementation for an unbounded number of IMCs, showing that these problems are EXPTIME-complete.

We discuss suitable notions of determinism for specifications, and show that for deterministic IMCs the syntactic refinement operators are complete with respect to model inclusion. Finally, we show that deciding consistency (emptiness) for an IMC is polynomial and that existence of common implementation can be established in polynomial time for any constant number of IMCs.

Contributions

- Describes weak refinement; a variant of the refinement relation considered by [46]
- Thorough, weak, and strong refinement coincides for deterministic Interval Markov Chains
- Checking thorough refinement is EXPTIME-complete
- Checking the existence of a common implementation between k Interval Markov Chains is EXPTIME-complete and in PTIME if k is fixed
- Checking consistency between two Interval Markov Chains is in PTIME

Publication history This paper has been accepted at *Journal of Logic and Algebraic Programming (JLAP)*. It is the journal version of the paper *Decision Problems for Interval Markov Chains* [61], that has been accepted at the *5th International Conference on Language and Automata Theory and Applications (LATA'11)*. An extended abstract based on the conference paper was accepted for presentation at the *22nd Nordic Workshop on Programming Theory (NWPT'10)*.

Paper B – *Constraint Markov Chains*

Notions of specification, implementation, satisfaction, and refinement, together with operators supporting stepwise design, constitute a specification theory. We construct

such a theory for Markov Chains (MCs) employing a new abstraction of a Constraint MC. Constraint MCs permit rich constraints on probability distributions and thus generalize prior abstractions such as Interval MCs. Linear (polynomial) constraints suffice for closure under conjunction (respectively parallel composition). This is the first specification theory for MCs with such closure properties. We discuss its relation to simpler operators for known languages such as probabilistic process algebra. Despite the generality, all operators and relations are computable.

Contributions

- The notions of conjunction and parallel composition are presented. The idea of using synchronizers, to obtain synchronized parallel composition, is introduced.
- Thorough, weak, and strong refinement coincides for deterministic Constraint Markov Chains
- Probabilistic Automata can be translated to Constraint Markov Chains for which weak refinement, strong refinement, and parallel composition coincides with simulation, probabilistic bisimulation, and parallel composition for the respective Probabilistic Automata.

Publication history This paper [62] has been accepted in *Theoretical Computer Science (TCS)*. It is a journal version of the paper *Composition Design Methodology with Constraint Markov Chains* [63], that has been accepted at the *7th International Conference on Quantitative Evaluation of SysTems (QEST'10)*.

Paper C – *New Results for Constraint Markov Chains*

This paper studies compositional reasoning theories for stochastic systems. A specification theory combines notions of specification and implementation with satisfaction and refinement relations, and a set of operators that together support stepwise design. One of the first behavioural specification theories introduced for stochastic systems is the one of Interval Markov Chains (IMCs), which are Markov Chains whose probability distributions are replaced by a conjunction of intervals. In this paper, we show that IMCs are not closed under conjunction, which gives a formal proof of a conjecture made in several recent works.

In order to leverage this problem, we suggested to work with *Constraint Markov Chains* (CMCs) that is another specification theory where intervals are replaced with general constraints. Contrary to IMCs, one can show that CMCs enjoy the good closure properties of a specification theory. In addition, we propose aggressive abstraction procedures for CMCs. Such abstractions can be used either to combat the state-space explosion problem, or to simplify complex constraints. In particular, one can show that, under some assumptions, the behavior of any CMC can be abstracted by an IMC.

Finally, we propose an algorithm for counter-example generation, in case a refinement of two CMCs does not hold. We present a tool that implements our results.

Implementing CMCs is a complex process and relies on recent advances made in decision procedures for theory of reals.

Contributions

- A formal proof that Interval Markov Chains are not closed under conjunction
- Notions of abstraction are introduced for Constraint Markov Chains
- The tool APAC is presented for Constraint Markov Chains, as well as the counter-example generation algorithm

Publication history This paper has been accepted in *Performance Evaluation (PEVA)*. It is, like paper B, a journal version of the paper *Composition Design Methodology with Constraint Markov Chains* [63].

Paper D – *Abstract Probabilistic Automata*

Probabilistic Automata (PAs) are a widely-recognized mathematical framework for the specification and analysis of systems with non-deterministic and stochastic behaviors. This paper proposes Abstract Probabilistic Automata (APAs), that is a novel abstraction model for PAs. In APAs uncertainty of the non-deterministic choices is modeled by may/must modalities on transitions while uncertainty of the stochastic behaviour is expressed by (underspecified) stochastic constraints. We have developed a complete abstraction theory for PAs, and also propose the first specification theory for them. Our theory supports both satisfaction and refinement operators, together with classical stepwise design operators. In addition, we study the link between specification theories and abstraction in avoiding the state-space explosion problem.

Contributions

- Abstract Probabilistic Automata combines techniques from Modal Transition Systems and Constraint Markov Chains
- Notions of abstraction are introduced for Abstract Probabilistic Automata
- Thorough, weak, and strong refinement coincides for deterministic Abstract Probabilistic Automata, by extending the same result for Interval Markov Chains and Constraint Markov Chains

Publication history This paper [64] has been accepted at the *12th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'11)*.

Paper E – *New Results on Abstract Probabilistic Automata*

Probabilistic Automata (PAs) are a recognized framework for modeling and analysis of nondeterministic systems with stochastic behavior. Recently, we proposed Abstract Probabilistic Automata (APAs)—an abstraction framework for PAs. In this paper, we discuss APAs over dissimilar alphabets, a determinisation operator, conjunction of non-deterministic APAs, and an APA-embedding of Interface Automata. We conclude introducing a tool for automatic manipulation of APAs.

Contributions

- A version of conjunction, allowing to combine non-deterministic Abstract Probabilistic Automata, is defined. This conjunction is the greatest lower bound wrt. the presented weak weak refinement
- A determinization algorithm is defined for Abstract Probabilistic Automata
- The notion of Abstract Probabilistic Interfaces is introduced
- The tool APAC is presented for Abstract Probabilistic Automata

Publication history This paper [65] has been accepted at the *11th International Conference on Application of Concurrency to System Design (ACSD'11)*.

Paper F – *APAC: a tool for reasoning about Abstract Probabilistic Automata*

We recently introduced Abstract Probabilistic Automata (APA), a new powerful abstraction formalism for probabilistic automata. Our theory is equipped with a series of aggressive abstraction techniques for state-space reduction as well as a specification theory for both logical and structural comparisons. This paper reports on the implementation of the approach in the Abstract Probabilistic Automata Checker toolset.

Contributions

- The tool APAC is presented

Publication history This paper [66] has been accepted as a tool paper at the *8th International Conference on Quantitative Evaluation of SysTems (QEST'11)*.

Paper A – Consistency and Refinement for Interval Markov Chains

Benoit Delahaye
INRIA/IRISA, Rennes

Kim G. Larsen
Aalborg University

Axel Legay
INRIA/IRISA, Rennes

Mikkel L. Pedersen
Aalborg University

Andrzej Wasowski
IT University, Copenhagen

1 Abstract

Interval Markov Chains (IMC), or Markov Chains with probability intervals in the transition matrix, are the base of a classic specification theory for probabilistic systems (Larsen and Jonsson, 1991). The standard semantics of IMCs assigns to a specification the set of all Markov Chains that satisfy its interval constraints. The theory then provides operators for deciding emptiness of conjunction and refinement (entailment) for such specifications.

In this paper we study complexity of several problems for IMCs, that stem from compositional modeling methodologies. In particular, we close the complexity gap for thorough refinement of two IMCs and for deciding the existence of a common implementation for an unbounded number of IMCs, showing that these problems are EXPTIME-complete.

We discuss suitable notions of determinism for specifications, and show that for deterministic IMCs the syntactic refinement operators are complete with respect to model inclusion. Finally, we show that deciding consistency (emptiness) for an IMC is polynomial and that existance of common implementation can be established in polynomial time for any constant number of IMCs.

2 Introduction

Interval Markov Chains (IMCs for short) extend Markov Chains, by allowing to specify intervals of possible probabilities on state transitions. IMCs have been introduced by Larsen and Jonsson [46] as a *specification* formalism—a basis for a stepwise-refinement-like modeling method, where initial designs are very abstract and underspecified, and then they are made continuously more precise, until they are concrete. Unlike richer specification models, such as Constraint Markov Chains [63], IMCs are difficult to use for compositional specification due to lack of basic modeling operators. To address this, we study complexity and algorithms for deciding consistency of conjunctive sets of IMC specifications.

Let us consider an example. Figure 1 presents a simple specification of a user of coffee machine. The model on the left hand side prescribes that a typical user orders coffee with milk with probability $x \in [0, 0.5]$ and black coffee with probability $y \in [0.2, 0.7]$ (customers also buy tea with probability $t \in [0, 0.5]$).

Jonsson and Larsen [46] have introduced refinement of such processes, but have not characterized its computational complexity. Refinement allows deciding whether one specification allows a subset of the probabilistic processes allowed by another one. We extend the work on refinement by classifying its complexity and characterizing it using structural coinductive algorithms in the style of simulation.

Consider the issue of combining multiple specifications of the same system. It turns out that conjunction of IMCs cannot be expressed as an IMC itself, due to a lack of expressiveness of intervals. We have recently shown this formally in a parallel work [67]. Here we illustrate this with an example. The right hand side model in Figure 1 presents a different view on the coffee service. The vendor of the machine delivers another specification, which prescribes that the machine is serviceable only if coffee (white or black) is ordered with some probability $z \in [0.4, 0.8]$ from among other beverages, otherwise it will run out of coffee powder too frequently, or the powder becomes too old. A conjunction of these two models would describe usage patterns compatible with this particular machine. Such a conjunction effectively requires that all the interval constraints are satisfied and that $z = x + y$ holds. However, the solution of this constraint is not described by an interval over x and y. This can be seen by pointing out an extremal point, which is not a solution, while all its coordinates take part in some solution. Say $x = 0$ and $y = 0.2$ violates the interval for z, while for each

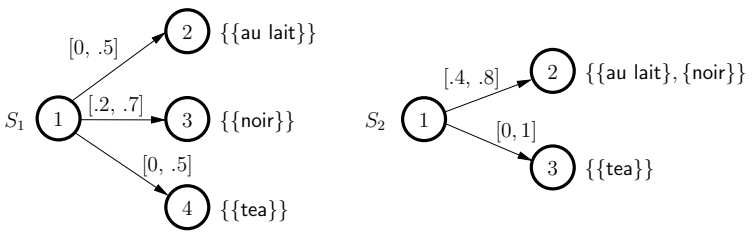


Figure 1: Two specifications of different aspects of a coffee service

of these two values it is possible to select another one in such a way that z's constraint is also held (for example $(x = 0, y = 0.4)$ and $(x = 0.2, y = 0.2)$). Thus the solution space is not an interval over x and y. This lack of closure properties for IMCs motivates us to address the problem of reasoning about conjunction, without constructing it — the, so called, common implementation problem.

In this paper we provide algorithms and complexities for thorough refinement, consistency, common implementation, and refinement of IMCs, in order to enable compositional modeling. We contribute the following new results:

- We define suitable notions of determinism for IMCs, and show that for deterministic IMCs *thorough refinement* (TR) coincides with two simulation-like preorders (the *weak refinement* and *strong refinement*), for which there exist co-inductive algorithms terminating in a polynomial number of iterations.
- In [46] a TR between IMCs is defined as an inclusion of their implementation sets. We show that the procedure for deciding TR given in [46] can be implemented in single exponential time. Furthermore, we provide a lower bound, concluding that TR is EXPTIME-complete. While the reduction from TR of modal transition systems [43] used to provide this lower bound is conceptually simple, it requires a rather involved proof of correctness, namely that it preserves sets of implementations in a sound and complete manner.
- A polynomial procedure for checking whether an IMC is *consistent* (C), i.e. it admits an implementation as a Markov Chain.
- An exponential procedure for checking whether k IMCs are consistent in the sense that they share a Markov Chain satisfying all—a *common implementation* (CI). We show that this problem is EXPTIME-complete.
- As a special case, we observe that CI is PTIME for any constant value of k. In particular, checking whether two specifications can be simultaneously satisfied, and synthesizing their shared implementation can be done in polynomial time.

The paper proceeds as follows. We begin by summarizing prior work on these and related problems, and surveying application areas for Interval Markov Chains (Section 3). In Section 4 we introduce the basic definitions. All results in subsequent sections are new and ours. In Section 5 we discuss deciding TR and other refinement procedures. We expand on the interplay of determinism and refinements in Section 6. The problems of C and CI are addressed in Section 7. We conclude by discussing the results in Section 8.

3 State of The Art

Besides IMCs, there exists many other specification formalisms for describing and analyzing stochastic systems; the list includes process algebras [29, 68] or logical frameworks [18]. A logical representation is suited for conjunction. The process algebraic

specifications tend to be well developed for parallel composition and efficient refinement checking. For example, it is not clear how one can synthesize a MC (an implementation) that satisfies two Probabilistic Computation Tree Logic formulas. Similarly, conjunction is usually not defined for process algebraic specifications. In this sense, IMCs situate themselves in the middle between logical and process algebraic models—one can reason about their common implementation and refinement.

In mathematics, the abstraction of Markov set-chains [69] lies very close to IMCs. The latter defines the intervals on the transition probabilities, while the former uses matrix intervals in the transition matrix space, which allows reasoning about the abstraction using linear algebra. Technically, a Markov set-chain is an explicit enumeration of all the implementations of an IMC. Markov set-chains have been, for instance, used to approximate dynamics of hybrid systems [70]. Arguably, they have a different objective and compositional reasoning operators have not been considered for them, so far.

IMCs have served the purpose of *abstraction* in model checking, where a concrete system is being soundly abstracted by a less precise system in order to prove the properties more easily [47, 48, 71, 72]. The main issues related to model checking of IMCs have recently been addressed in [48].

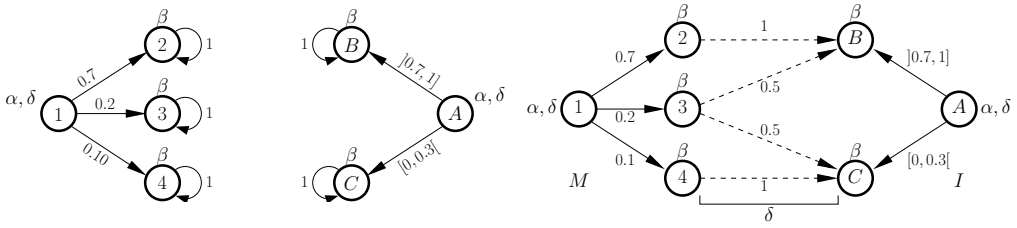
As we already stated, IMCs are not expressive enough to represent many artifacts of compositional design. In [63], we have presented Constraint Markov Chains (CMC) a specification model that, contrary to IMCs, is closed under composition and conjunction. While more expressive than IMCs, CMCs are not an immediate and universal replacement for IMCs, given that complexity of decision procedures for them is much higher. IMCs remain relevant, whenever parallel composition is not required in the application, or when they are used as a coarse abstraction (for example) for CMCs.

For functional analysis of discrete-time non-probabilistic systems, the theory of Modal Transition Systems (MTS) [35, 42] provides a specification formalism supporting refinement, conjunction and parallel composition. Earlier we have obtained EXPTIME-completeness both for the corresponding notion of CI [50] and of TR [43] for MTSs. In [46] it is shown that IMCs properly contain MTSs, which puts our new results in a somewhat surprising light: in the complexity theoretic sense, and as far as CI and TR are considered, the generalization of modalities by probabilities does come for free. A recent overview of research on (discrete) modal specifications is available in [73].

4 Background

We shall now introduce the basic definitions used throughout the paper. In the following we will write $\mathsf{Intervals}_{[0,1]}$ for the set of all closed, half-open and open intervals included in $[0, 1]$.

A Markov Chain (sometimes MC in short) is a tuple $C = \langle P, p_0, \pi, A, V_C \rangle$, where P is a set of states containing the initial state p_0, A is a set of atomic propositions, $V_C : P \to 2^A$ is a state valuation labeling states with propositions, and $\pi : P \to \mathsf{Distr}(P)$ is a probability distribution assignment such that $\sum_{p' \in P} \pi(p)(p') = 1$ for all $p \in P$.



(a) A Markov Chain M **(b)** An IMC I **(c)** An example of satisfaction relation

Figure 2: Markov Chain, Interval Markov Chain and satisfaction relation

The probability distribution assignment is the only component that is relaxed in IMCs:

Definition 1 (Interval Markov Chain)**.** *An* Interval Markov Chain *is a tuple* $I = \langle Q, q_0, \varphi, A, V_I \rangle$*, where* Q *is a finite set of states containing the initial state* q_0*,* A *is a set of atomic propositions,* $V_I : Q \to 2^A$ *is a state valuation, and* $\varphi : Q \to (Q \to \mathsf{Intervals}_{[0,1]})$*, which for each* $q \in Q$ *and* $q' \in Q$ *gives an interval of probabilities.*

Instead of a distribution, as in MCs, in IMCs we have a function mapping elementary events (target states) to intervals of probabilities. We interpret this function as a constraint over distributions. This is expressed in our notation as follows. Given a state $q \in Q$ and a distribution $\sigma \in \mathsf{Distr}(Q)$, we say that $\sigma \in \varphi(q)$ iff $\sigma(q') \in \varphi(q)(q')$ for all $q' \in Q$. Occasionally, it is convenient to think of a Markov Chain as an IMC, in which all probability intervals are closed point intervals.

We visualize IMCs as automata with intervals on transitions. As an example, consider the IMC in Figure 2b. It has two outgoing transitions from the initial state A. No arc is drawn between states if the probability is zero (or, more precisely, the interval is $[0, 0]$), so in the example there is zero probability of going from state A to A, or from B to C, etc. Otherwise, the probability distribution over successors of A is constrained to fall into $]0.7, 1]$ and $[0, 0.3]$ for B and C respectively. States B and C have valuation β, whereas state A has valuation α, δ. Please observe that Figure 2a presents a Markov Chain using the same convention, modulo the intervals. Remark that our formalism does not allow "sink states", i.e. states with no outgoing transition. However, in order to avoid clutter in the figures, we sometimes represent states with no outgoing transitions. They must be interpreted as states with a self-loop with a closed point interval consisting of probability 1.

A *satisfaction* relation establishes compatibility of Markov Chains (implementations) and IMCs (specifications). The original definition of satisfaction between MCs and IMCs was presented in [30, 46]. We use a slightly modified, but strictly equivalent definition using a concept of *correspondence functions*:

Definition 2 (Satisfaction)**.** *Let* $C = \langle P, p_0, \pi, A, V_C \rangle$ *be a MC and let* $I = \langle Q, q_0, \varphi, A, V_I \rangle$ *be an IMC. A relation* $\mathcal{R} \subseteq P \times Q$ *is called a satisfaction relation if whenever* $p \mathcal{R} q$ *then*

- *Their valuation sets agree:* $V_C(p) = V_I(q)$
- *There exists a correspondence function* $\delta : P \to (Q \to [0,1])$ *such that*
  1. *For all* $p' \in P$, *if* $\pi(p)(p') > 0$ *then* $\delta(p')$ *defines a distribution on* Q,
  2. $\sum_{p' \in P} \pi(p)(p')\delta(p')(q') \in \varphi(q)(q')$ *for all* $q' \in Q$, *and*
  3. *if* $\delta(p')(q') > 0$, *then* $p' \mathcal{R} q'$.

We write $C \models I$ iff there exists a satisfaction relation containing (p_0, q_0). C is an *implementation* of I. The set of implementations of I is written $[\![I]\!]$. Figure 2c presents an example of satisfaction on states 1 and A. The correspondence function is specified using labels on the dashed arrows i.e. the probability mass going from state 1 to 3 is distributed to state B and C with half going to each.

We will say that a state q of an IMC is *consistent* if its interval constraint $\varphi(q)$ is satisfiable, i.e., there exists a distribution $\sigma \in \mathsf{Distr}(Q)$ satisfying $\varphi(q)$. Obviously, for a given IMC, it is sufficient that all its states are consistent in order to guarantee that the IMC is consistent itself—there exists a Markov Chain satisfying it. We discuss the problem of establishing consistency in a sound and complete manner in Section 7.

There are three known ways of defining refinement for IMCs: the strong refinement (introduced as *simulation* in [46]), weak refinement (introduced under the name of *probabilistic simulation* in [48]), and thorough refinement (introduced as *refinement* in [46]). We will recall their formal definitions:

Definition 3 (Strong Refinement)**.** *Let* $I_1 = \langle Q, q_0, \varphi_1, A, V_1 \rangle$ *and* $I_2 = \langle S, s_0, \varphi_2, A, V_2 \rangle$ *be two IMCs. A relation* $\mathcal{R} \subseteq Q \times S$ *is called a* strong refinement relation *if whenever* $q \mathcal{R} s$, *then*

- *Their valuation sets agree:* $V_1(q) = V_2(s)$ *and*
- *There exists a correspondence function* $\delta : Q \to (S \to [0,1])$ *such that for all* $\sigma \in \mathsf{Distr}(Q)$, *if* $\sigma \in \varphi_1(q)$, *then*
  1. *for each* $q' \in Q$ *such that* $\sigma(q') > 0$, $\delta(q')$ *is a distribution on* S,
  2. *for all* $s' \in S$, *we have* $\sum_{q' \in Q} \sigma(q')\delta(q')(s') \in \varphi_2(s)(s')$, *and*
  3. *for all* $q' \in Q$ *and* $s' \in S$, *if* $\delta(q')(s') > 0$, *then* $q' \mathcal{R} s'$.

I_1 *strongly refines* I_2, *written* $I_1 \leq_{\mathsf{S}} I_2$, *iff there exists a strong refinement relation containing* (q_0, s_0).

A strong refinement relation requires existence of a single correspondence, which witnesses satisfaction for any resolution of probability constraint over successors of q and s. Figure 3a illustrates such a correspondence between states A and α of two IMCs. The correspondence function is given by labels on the dashed lines. It is easy to see that regardless of how the probability constraints are resolved the correspondence function distributes the probability mass in a fashion satisfying α.

We now recall the notion of *weak refinement*, first introduced in [48] under the name of probabilistic simulation.

Definition 4 (Weak Refinement)**.** *Let $I_1 = \langle Q, q_0, \varphi_1, A, V_1\rangle$ and $I_2 = \langle S, s_0, \varphi_2, A, V_2\rangle$ be two IMCs. A relation $\mathcal{R} \subseteq Q \times S$ is called a* weak refinement relation *if whenever $q\,\mathcal{R}\,s$, then*

- *Their valuation sets agree: $V_1(q) = V_2(s)$*
- *For each $\sigma \in \mathsf{Distr}(Q)$ such that $\sigma \in \varphi_1(q)$, there exists a correspondence function $\delta : Q \to (S \to [0,1])$ such that*
  1. *For each $q' \in Q$ such that $\sigma(q') > 0$, $\delta(q')$ is a distribution on S,*
  2. *for all $s' \in S$, we have $\sum_{q' \in Q} \sigma(q')\delta(q')(s') \in \varphi_2(s)(s')$, and*
  3. *for all $q' \in Q$ and $s' \in S$, if $\delta(q')(s') > 0$, then $q'\,\mathcal{R}\,s'$.*

I_1 weakly refines I_2, written $I_1 \leq_{\mathsf{W}} I_2$, iff there exists a weak refinement relation containing (q_0, s_0).

The weak refinement between two states requires that, for any resolution of probability constraint over successors in I_1, there exists a correspondence function which witnesses satisfaction of I_2. Thus the weak refinement achieves the weakening by swapping the order of quantifications. Figure 3b illustrates such a correspondence between states A and α of another two IMCs. Here, x stands for a value in $[0.2, 1]$ (arbitrary choice of probability of going to state C from A). Notably, for each choice of x, there exists $p \in [0,1]$ such that $px \in [0, 0.6]$ and $(1-p)x \in [0.2, 0.4]$. Remark that strong refinement naturally implies weak refinement. Indeed, if there exists a single correspondence function witnessing satisfaction for any resolution of the constraints, then there exists a correspondence function for each resolution of the constraints.

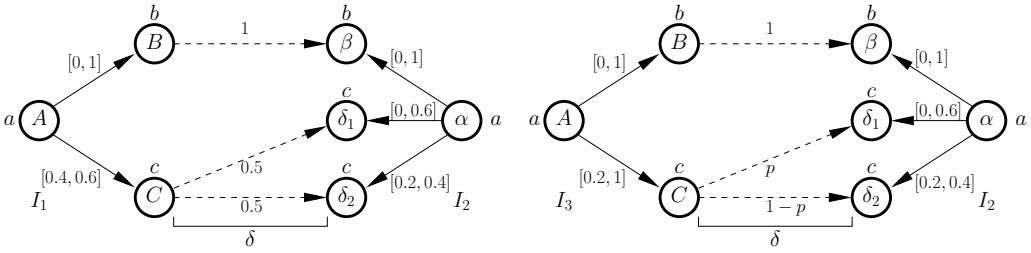
Finally, we introduce the thorough refinement as defined in [46]:

Definition 5 (Thorough Refinement)**.** *IMC I_1* thoroughly refines *IMC I_2, written $I_1 \leq_{\mathsf{T}} I_2$, iff each implementation of I_1 implements I_2: $[\![I_1]\!] \subseteq [\![I_2]\!]$*

Thorough refinement is the ultimate refinement relation for any specification formalism, as it is based on the semantics of the models.

5 Refinement Relations

We will now compare the expressiveness of the refinement relations. It is not hard to see that both strong and weak refinements soundly approximate the thorough refinement (since they are transitive and degrade to satisfaction if the left argument is a Markov Chain). The converse does not hold. We will now discuss procedures to compute weak and strong refinements, and then compare the granularity of these relations, which will lead us to procedures for computing thorough refinement. Observe that all three



(a) Illustration of a strong refinement relation between an IMC I_1 and an IMC I_2

(b) A weak refinement relation between an IMC I_3 and an IMC I_2; p is a parameter

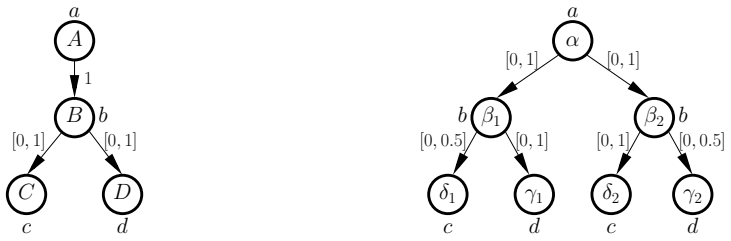
Figure 3: Illustration of strong and weak refinement relations

refinement are decidable as they only rely on the first order theory of real numbers. In concrete cases below the calculations can be done more efficiently due to convexity of solution spaces for interval constraints.

Weak and Strong Refinement Consider two IMCs $I_1 = \langle P, o_1, \varphi_1, A, V_1\rangle$ and $I_2 = \langle Q, o_2, \varphi_2, A, V_2\rangle$. Informally, checking whether a given relation $\mathcal{R} \subseteq P \times Q$ is a weak refinement relation reduces to checking, for each pair $(p, q) \in \mathcal{R}$, whether the following formula is true: $\forall \pi \in \varphi_1(p)\ \exists \delta : P \to (Q \to [0, 1])$ such that $\pi\delta$ satisfies a system of linear equations / inequalities. Since the set of distributions satisfying $\varphi_1(p)$ is convex, checking such a system is exponential in the number of variables, here $|P||Q|$. As a consequence, checking whether a relation on $P \times Q$ is a weak refinement relation is exponential in $|P||Q|$. For strong refinement relations, the only difference appears in the formula that must be checked: $\exists \delta : P \to (Q \to [0, 1])$ such that $\forall \pi \in \varphi_1(p)$, we have that $\pi\delta$ satisfies a system of linear equations / inequalities. Therefore, checking whether a relation on $P \times Q$ is a strong refinement relation is also exponential in $|P||Q|$.

Deciding whether weak (strong) refinement holds between I_1 and I_2 can be done in the usual coinductive fashion by considering the total relation $P \times Q$ and successively removing all the pairs that do not satisfy the above formulae. The refinement holds iff the relation we reach contains the pair (o_1, o_2). The algorithm will terminate after at most $|P||Q|$ iterations. This gives an upper bound on the complexity to establish strong and weak refinements: a polynomial number of iterations over an exponential step. This upper bound may be loose. One could try to reuse techniques for non-stochastic systems [74] in order to reduce the number of iterations. This is left to future work.

Granularity In [46] an informal statement is made that the strong refinement is strictly stronger (finer) than the thorough refinement: $(\leq_{\mathsf{T}}) \supsetneq (\leq_{\mathsf{S}})$. In [48], the weak refinement is introduced without discussing its relations to neither the strong nor the thorough refinement. The following theorem resolves all open issues in relations between the three:



(a) IMC I_4 **(b)** IMC I_5

Figure 4: IMCs I_4 and I_5 such that I_4 thoroughly but not weakly refines I_5

Theorem 1. *The thorough refinement is strictly weaker than the weak refinement, which is strictly weaker than the strong refinement:* $(\leq_{\mathsf{T}}) \supsetneq (\leq_{\mathsf{W}}) \supsetneq (\leq_{\mathsf{S}})$.

Proof. First, remark that weak refinement implies thorough refinement. Indeed, weak refinement is transitive and degrades to satisfaction when its left argument is a Markov chain. Thus it is equivalent to say that a MC M satisfies an IMC I and that $M \preceq_W I$. If furthermore $I \preceq_W I'$, then, by transitivity, we obtain $M \preceq_W I'$, which is equivalent to $M \models I'$. As a consequence, if $I \preceq_W I'$, then for all MC M such that $M \models I$, it holds that $M \models I'$, i.e. $[\![I]\!] \subseteq [\![I']\!]$.

We now consider the two inequalities separately.

1. Case 1: $(\leq_{\mathsf{T}}) \supsetneq (\leq_{\mathsf{W}})$. Figure 4 proposes two IMCs I_4 and I_5, such that I_4 thoroughly but not weakly refines I_5. Indeed, let $M = \langle Q, q_0, \pi, \{a, b, c, d\}, V_M \rangle$ be an implementation of I_4 and $\mathcal{R}$ a corresponding satisfaction relation. Let $P \subseteq Q$ be the set of states of M satisfying B. Consider a state $p \in P$. Let $\pi^C(p) = \sum_{\{q \in Q \mid q \,\mathcal{R}\, C\}} \pi(p)(q)$ and $\pi^D(p) = \sum_{\{q \in Q \mid q \,\mathcal{R}\, D\}} \pi(p)(q)$. Since $p \,\mathcal{R}\, B$, we have that $\pi^C(p) + \pi^D(p) = 1$. Let $P_1 \subset P$ be the set of states of M such that $\pi^C(p) \leq 0.5$ and let $P_2 \subset P$ be the set of states of M such that $\pi^D(p) < 0.5$. Obviously, we have $P = P_1 \cup P_2$ and $P_1 \cap P_2 = \emptyset$. By construction, the states in P_1 will satisfy β_1 and the states in P_2 will satisfy β_2. We now build a satisfaction relation $\mathcal{R}'$ such that, for all $q \in M$, if $q \,\mathcal{R}\, A$, then $q \,\mathcal{R}\, \alpha$; if $q \in P_1$, then $q \,\mathcal{R}'\, \beta_1$; if $q \in P_2$, then $q \,\mathcal{R}'\, \beta_2$; if $q \,\mathcal{R}\, C$, then $q \,\mathcal{R}'\, \delta_1$ and $q \,\mathcal{R}'\, \delta_2$; and if $q \,\mathcal{R}\, D$ then $q \,\mathcal{R}'\, \gamma_1$ and $q \,\mathcal{R}'\, \gamma_2$. By construction, $\mathcal{R}'$ is a satisfaction relation, and M is an implementation of I_5. Thus, $[\![I_4]\!] \subseteq [\![I_5]\!]$. However, it is not possible to define a weak refinement relation between I_4 and I_5: obviously, B can neither refine β_1 nor β_2.

2. Case 2: $(\leq_{\mathsf{W}}) \supsetneq (\leq_{\mathsf{S}})$. In Figure 3b, we propose two IMCs, I_3 and I_2 such that I_3 weakly but not strongly refines I_2. State A weakly refines state α: Given a value x for the transition $A \to C$, we can split it in order to match both transitions

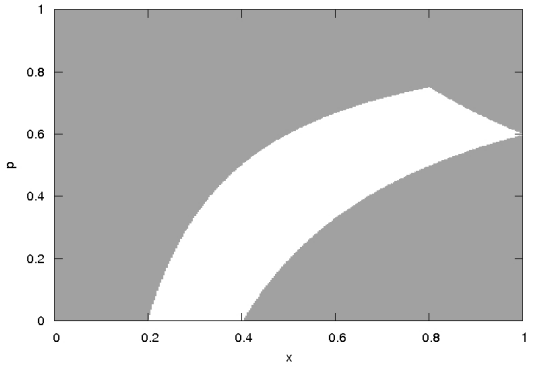


Figure 5: Solutions (in white) of the system of inequalities (1)

$\alpha \xrightarrow{px} \delta_1$ and $\alpha \xrightarrow{(1-p)x} \delta_2$. Define $\delta(C)(\delta_1) = p$ and $\delta(C)(\delta_2) = (1-p)$, with

$$p = \begin{cases} 0 & \text{if } 0.2 \leq x \leq 0.4 \\ \frac{x-0.3}{x} & \text{if } 0.4 < x < 0.8 \\ 0.6 & \text{if } 0.8 \leq x \end{cases}$$

δ_1 is a correspondence function witnessing a weak refinement relation between A and α. Consider the following parametric inequalities, where p is the variable and x the parameter.

$$\begin{aligned} xp &\leq 0.6 \\ x(1-p) &\leq 0.4 \\ x(1-p) &\geq 0.2 \end{aligned} \tag{1}$$

Suppose that a strong refinement relation $\mathcal{R}$ exists between I_3 and I_2. Then the correspondence function witnessing $A \,\mathcal{R}\, \alpha$ should be similar to the one given above, where p would be a constant solution of the system of inequalities (1). However, one can see from the solutions of this system of inequalities, which are graphically represented in Figure 5, that there exists no value of p satisfying (1) for all x.

□

Deciding Thorough Refinement As weak and strong refinements are strictly stronger than thorough refinement, it is interesting to investigate the complexity of deciding TR. In [46] a procedure computing TR is given, albeit without a complexity class. We now establish the complexity of this procedure, closing the problem:

Theorem 2. *The decision problem TR of establishing whether there exists a thorough refinement between two given IMCs is EXPTIME-complete.*

The proofs for both the upper and the lower bounds rely on a series of results that are presented in the rest of this section.

The upper bound. The upper-bound is shown by analyzing the complexity of the algorithm presented in [46]. For the sake of completeness, and in order to clarify several typesetting inaccuracies of the original presentation, we quote the construction of [46] below and subsequently analyze its complexity:

Definition 6 (Subset simulation)**.** *Let $I_1 = \langle Q, q_0, \varphi_Q, A, V_Q\rangle$ and $I_2 = \langle P, p_0, \varphi_P, A, V_P\rangle$ be IMCs. A total relation $\mathcal{R} \subseteq Q \times 2^P$ is a* subset-simulation *iff for each state $q \in Q$:*

1. *$q\,\mathcal{R}\,T$ implies $V_Q(q) = V_P(t)$ for all $t \in T$*
2. *For each probability distribution $\pi_Q \in \varphi_Q(q)$ and each correspondence function $\delta_Q : Q \to (2^P \to [0,1])$ such that $\mathsf{support}(\delta_Q) \subseteq \mathcal{R}$, there exists a set T such that $q\,\mathcal{R}\,T$ and for each $t \in T$, there exists a probability distribution $\pi_P \in \varphi_P(t)$ and a correspondence function $\delta_P : P \to (2^P \to [0,1])$ such that*
   (a) *if $\delta_P(t')(T') > 0$, then $t' \in T'$, and*
   (b) *for all $T' \in 2^P$, we have*

$$\sum_{q' \in Q} \pi_Q(q')\delta_Q(q')(T') = \sum_{p' \in P} \pi_P(p')\delta_P(p')(T').$$

Intuitively, this relation associates to every state q of I_1 a sample of sets of states $(T_1, \ldots, T_k)$ of I_2 that are "compatible" with q. Then, for each admissible redistribution δ of the successor states of q, it states that there exists one of the sets T_i such that for each of its states t', there is a redistribution γ of the successor states of t' that is compatible with δ. In [46] it is shown that the existence of a subset-simulation between two IMCs I_1 and I_2 is equivalent to thorough refinement between them. We now propose an example to illustrate the subset simulation algorithm presented above.

Example 1. *Consider the IMCs $I_4 = \langle\{A, B, C, D\}, A, \varphi_4, \{a, b, c, d\}, V_4\rangle$ and $I_5 = \langle\{\alpha, \beta_1, \beta_2, \delta_1, \delta_2, \gamma_1, \gamma_2\}, \alpha, \varphi_5, \{a, b, c, d\}, V_5\rangle$ given in Figure 4. They are such that I_4 thoroughly but not weakly refines I_5 (c.f. proof of Theorem 1). Since thorough refinement holds, we can exhibit a subset simulation $\mathcal{R} \subseteq P \times 2^Q$ between I_4 and I_5: Let $\mathcal{R} = \{(A, \{\alpha\}), (B, \{\beta_1\}), (B, \{\beta_2\}), (C, \{\delta_1, \delta_2\}), (D, \{\gamma_1, \gamma_2\})\}$ be this subset simulation. We illustrate the unfolding of $\mathcal{R}$ for states A and B of I_4. The rest is left to the reader.*

Consider state A of I_4.

1. *We have $A\,\mathcal{R}\{\alpha\}$, and $V_4(A) = a = V_5(\alpha)$.*

2. *The only distribution $\pi \in \varphi_4(A)$ is such that $\pi(B) = 1$. Let for example $\Delta^1 \in [0,1]^{4\times 2^7}$ be the correspondence matrix such that $\Delta^1_{B,\{\beta_1\}} = 1/2$ and $\Delta^1_{B,\{\beta_2\}} = 1/2$. Let $\{\alpha\}$ be the set such that $A\,\mathcal{R}\{\alpha\}$. Let ρ be the distribution on Q such that $\rho(\beta_1) = \rho(\beta_2) = 1/2$. ρ is indeed in $\varphi_5(\alpha)$. Let $\Delta^2 \in [0,1]^{7\times 2^7}$ be the correspondance matrix such that $\Delta^2_{\beta_1,\{\beta_1\}} = 1$ and $\Delta^2_{\beta_2,\{\beta_2\}} = 1$. It is then obvious that*

   (a) *for all t and T, if $\Delta^2_{t,T} > 0$, then $t \in T$;*

   (b) *$\pi\Delta^1 = \rho\Delta^2$ holds.*

Consider state B of I_4.

1. *We have $B\,\mathcal{R}\{\beta_1\}$ and $B\,\mathcal{R}\{\beta_2\}$. It holds that $V_4(B) = b = V_5(\beta_1) = V_5(\beta_2)$.*

2. *Consider a distribution $\pi \in \varphi_4(B)$ (for example such that $\pi(C) < 1/2$). Let Δ^1 be an admissible correspondance matrix. We must have $\Delta^1_{C,\{\delta_1,\delta_2\}} = 1$ and $\Delta^1_{D,\{\gamma_1,\gamma_2\}} = 1$. Consider $\{\beta_1\}$ the set such that $B\,\mathcal{R}\{\beta_1\}$ (if $\pi(C) > 1/2$ then pick up $\{\beta_2\}$ instead). Let ρ be the distribution such that $\rho(\delta_1) = \pi(C)$ and $\rho(\gamma_1) = \pi(D)$. Since $\pi(C) < 1/2$, we have $\rho \in \varphi_5(\beta_1)$. Let Δ^2 be a correspondance matrix such that $\Delta^2_{\delta_1,\{\delta_1,\delta_2\}} = 1$ and $\Delta^2_{\gamma_1,\{\gamma_1,\gamma_2\}} = 1$. It is obvious that*

   (a) *for all t and T, if $\Delta^2_{t,T} > 0$, then $t \in T$;*

   (b) *$\pi\Delta^1 = \rho\Delta^2$ holds.*

The rest of the unfolding is obvious, and $\mathcal{R}$ is thus a subset simulation.

The existence of a subset simulation between two IMCs is decided using a standard co-inductive fixpoint calculation. The algorithm works as follows: first consider the total relation and check whether it is a subset-simulation. Then refine it by removing violating pairs of states, and check again until a fixpoint is reached (it becomes a subset-simulation or it is empty). Checking whether a given relation is a subset simulation has a single exponential complexity. Checking the second condition in the definition can be done in single exponential time by solving polynomial constraints with fixed quantifiers for each pair (q, T) in the relation. There are at most $|Q|2^{|P|}$ such pairs, which gives a single exponential time bound for the cost of one iteration of the fixpoint loop. There are at most $|Q|2^{|P|}$ elements in the total relation and at least one is removed in an iteration, which gives $O(|Q|2^{|P|})$ as the bound on the number of iterations. Since a polynomial of two exponentials is still an exponential, we obtain a single exponential time for running time of this computation.

Remark 1. *Summarizing, all three refinements are in EXPTIME. Still, weak refinement seems easier to check than thorough. For TR the number of iterations on the state-space of the relation is exponential while it is only polynomial for the weak refinement. Also, the constraint solved at each iteration involves a single quantifier alternation for the weak, and three alternations for the thorough refinement.*

The Lower Bound. The lower bound of Theorem 2 is shown by a polynomial reduction of the thorough refinement problem for modal transition systems to TR of IMCs. The former problem is known to be EXPTIME-complete [43].

A modal transition system (an MTS in short) [42] is a tuple $M = (S, s_0, A, \rightarrow, \dashrightarrow)$, where S is the set of states, s_0 is the initial state, and $\rightarrow \subseteq S \times A \times S$ are the transitions that *must* be taken and $\dashrightarrow \subseteq S \times A \times S$ are the transitions that *may* be taken. In addition, it is assumed that $(\rightarrow) \subseteq (\dashrightarrow)$.

A modal transition system $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ refines another modal transition system $N = (T, t_0, A, \rightarrow, \dashrightarrow)$ iff there exists a refinement relation $R \subseteq S \times T$ containing (s_0, t_0) such that if $(s, t) \in R$, then

1. whenever $t \xrightarrow{a} t'$ then also $s \xrightarrow{a} s'$ for some $s' \in S$ and $(s', t') \in R$
2. whenever $s \overset{a}{\dashrightarrow} s'$ then also $t \overset{a}{\dashrightarrow} t'$ for some $t' \in T$ and $(s', t') \in R$

A labelled transition system *implements* a MTS if it refines it in the above sense. Thorough refinement of MTSs is defined as inclusion of implementation sets, analogously to IMCs.

We now describe a translation of MTSs into IMCs which preserves implementations. We assume we only work with modal transition systems that have no deadlock-states, in the sense that each state has at least one outgoing must transition. This assumption is needed to avoid dealing with inconsistent states in the corresponding IMC. We first present a transformation that takes any two MTS and transforms them into MTS without deadlocks, preserving the notion of thorough refinement between them.

Let $M = \langle S, s_0, A, \rightarrow, \dashrightarrow \rangle$ be a MTS. Let $\perp \notin A$ be a new action variable, and $q \notin S$ be a new state variable. Define a new MTS $M_\perp = \langle S \cup \{q\}, s_0, A \cup \{\perp\}, \rightarrow_\perp, \dashrightarrow_\perp \rangle$ as follows: for all $s, s' \in S$ and $a \in A$, $s \xrightarrow{a}_\perp s' \iff s \xrightarrow{a} s'$ and $s \overset{a}{\dashrightarrow}_\perp s' \iff s \overset{a}{\dashrightarrow} s'$. Add the following transitions: for all $s \in S \cup \{q\}$, $s \xrightarrow{\perp}_\perp q$ and $s \overset{\perp}{\dashrightarrow}_\perp q$. In this way, every state of $M_\perp$ has at least one must outgoing transition. Moreover, it is trivial to see that this transformation preserves the notion of thorough refinement. This is stated in the following theorem:

Theorem 3. *Let M and M' be two MTS. If $\perp$ is in neither of their sets of actions, we have $[\![M]\!] \subseteq [\![M']\!] \iff [\![M_\perp]\!] \subseteq [\![M'_\perp]\!]$.*

Finally, we can safely suppose that all the MTS we consider in the rest of the section have no deadlocks.

We now describe an implementation preserving translation of MTSs into IMCs. The IMC $\widehat{M}$ corresponding to a MTS M is defined by the tuple $\widehat{M} = \langle Q, q_0, A \cup \{\epsilon\}, \varphi, V \rangle$ where $Q = S \times (\{\epsilon\} \cup A)$, $q_0 = (s_0, \epsilon)$, for all $(s, x) \in Q$, $V((s, x)) = \{x\}$ and φ is defined as follows : for all $t, s \in S$ and $b, a \in (\{\epsilon\} \cup A)$, $\varphi((t, b))((s', a)) =]0, 1]$ if $t \xrightarrow{a} s$; $\varphi((t, b))((s', a)) = [0, 0]$ if $t \not\overset{a}{\dashrightarrow} s$; and $\varphi((t, b))((s', a)) = [0, 1]$ otherwise. The encoding is illustrated in Figure 6.

We first state two lemmas that will be needed to prove the main theorem of the section: the encoding presented above reduces the problem of checking thorough refinement on modal transition systems to checking thorough refinement on IMCs.

(a) A MTS M **(b)** The IMC $\widehat{M}$

Figure 6: An example of the translation from Modal Transition Systems to IMCs

Lemma 4. *Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. We have $I \models M \Rightarrow [\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$.*

Proof. We first recall the definition of a satisfaction relation for MTS: Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. The implementation I satisfies the MTS M, written $I \models M$, iff there exists a relation $\mathcal{R} \subseteq S_I \times S$ such that

1. $s_0^I \,\mathcal{R}\, s_0$
2. Whenever $s_I \,\mathcal{R}\, s$, we have
   (a) For all $a \in A$, $s_I' \in S_I$, $s_I \xrightarrow{a} s_I'$ in I implies that there exists $s' \in S$ such that $s \overset{a}{\dashrightarrow} s'$ in M and $s_I' \,\mathcal{R}\, s'$.
   (b) For all $a \in A$, $s' \in S$, $s \xrightarrow{a} s'$ in M implies that there exists $s_I' \in S_I$ such that $s_I \xrightarrow{a} s_I'$ in M and $s_I' \,\mathcal{R}\, s'$.

Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. Let $\widehat{M} = \langle Q, q_0, A \cup \{\epsilon\}, \varphi, V\rangle$ and $\widehat{I} = \langle Q_I, (s_0^I, \epsilon), A \cup \{\epsilon\}, \varphi_I, V_I\rangle$ be the IMCs defined as above.
Suppose that $I \models M$. By definition, there exists a satisfaction relation for MTS $\mathcal{R} \subseteq S_I \times S$ such that $s_0^I \,\mathcal{R}\, s_0$. We show that $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$.

Let $T = \langle Q_T, p_0, \pi^T, V_T, A\rangle$ be an MC such that $T \in [\![\widehat{I}]\!]$. By definition, there exists a satisfaction relation for IMCs $\mathcal{R}_1 \subseteq Q_T \times Q_I$ such that $p_0 \,\mathcal{R}_1 (s_0^I, \epsilon)$. Define the new relation $\mathcal{R}_2 \subseteq Q_T \times Q$ such that $p \,\mathcal{R}_2 (s, x)$ iff there exists $s_I \in S_I$ such that $p \,\mathcal{R}_1 (s_I, x)$ and $s_I \,\mathcal{R}\, s$. We show that $\mathcal{R}_2$ is a satisfaction relation between T and $\widehat{M}$.

Let p, s, s_I, x be such that $p \,\mathcal{R}_1 (s_I, x)$ and $s_I \,\mathcal{R}\, s$, i.e. $p \,\mathcal{R}_2 (s, x)$. If $x \neq \bot$, we have

1. Since $p \,\mathcal{R}_1 (s_I, x)$, we have $V_T(p) = V_I((s_I, x)) = \{x\}$. Thus $V_T(p) = V((s, x)) = \{x\}$.
2. Let $\delta^1 \in \mathsf{Distr}(Q_T \times Q_I)$ be the probability distribution witnessing $p \,\mathcal{R}_1 (s_I, x)$, and let $\delta^2 \in \mathsf{Distr}(Q_T \times Q)$ be the correspondence matrix such that for all $p' \in$

Q_T, $s' \in S$ and $y \in A$, if $\{s'_I \in S_I \mid s'_I \mathcal{R}\, s'\} \neq \emptyset$ and $s \overset{y}{\dashrightarrow} s'$, then

$$\delta^2(p', (s', y)) = \sum_{\{s'_I \in S_I \mid s'_I \mathcal{R}\, s'\}} \frac{\delta^1(p', (s'_I, y))}{|\{s'' \in S \mid s'_I \mathcal{R}\, s'' \text{ and } s \overset{y}{\dashrightarrow} s''\}|};$$

Otherwise, $\delta^2(p', (s', y)) = 0$.

Recap that we suppose that all must transitions are also may transitions. The definition above potentially gives a non-zero value to $\delta^2(p', (s', y))$ if there exists a may (or must) transition from s to s' in S labelled with y and if there exists a state s'_I in I such that $s'_I \mathcal{R}\, s'$.

Let $p' \in Q_T$. We prove that $\sum_{(s',y)} \delta_2(p', (s', y)) = \pi^T(p)(p')$: By definition of δ^1, we have $\sum_{(s'_I,y)} \delta^1(p', (s'_I, y)) = \pi^T(p)(p')$.

$$\sum_{(s',y)} \delta^2(p', (s', y)) =$$

$$\sum_{\{(s',y) \mid \exists s'_I,\ s'_I \mathcal{R}\, s' \text{ and } s \overset{y}{\dashrightarrow} s'\}} \sum_{\{s'_I \mid s'_I \mathcal{R}\, s'\}} \frac{\delta^1(p', (s'_I, y))}{|\{s'' \in S \mid s'_I \mathcal{R}\, s'' \text{ and } s \overset{y}{\dashrightarrow} s''\}|}.$$

Clearly, for all (s'_I, y) such that $\delta^1(p', (s'_I, y)) > 0$, the term

$$\frac{\delta^1(p', (s'_I, y))}{|\{s'' \in S \mid s'_I \mathcal{R}\, s'' \text{ and } s \overset{y}{\dashrightarrow} s''\}|}$$

will appear exactly $|\{s'' \in S \mid s'_I \mathcal{R}\, s'' \text{ and } s \overset{y}{\dashrightarrow} s''\}|$ times in the expression above. As a consequence, $\sum_{(s',y)} \delta^2(p', (s', y)) = \sum_{(s'_I,y)} \delta^1(p', (s'_I, y)) = \pi^T(p)(p')$.

Moreover, we show that for all $(s', y) \in Q$, that

$$\sum_{p' \in Q^T} \delta^2(p', (s', y)) \in \varphi((s, x)(s', y)).$$

By construction, $\varphi((s, x)(s', y))$ is either $\{0\}$, $[0, 1]$ or $]0, 1]$. We will thus prove that (a) if $\sum_{p' \in Q^T} \delta^2(p', (s', y)) > 0$, then $\varphi((s, x)(s', y)) \neq \{0\}$; and (b) if $\varphi((s, x)(s', y)) =]0, 1]$, then $\sum_{p' \in Q^T} \delta^2(p', (s', y)) > 0$.

(a) Suppose $\sum_{p' \in Q^T} \delta^2(p', (s', y)) > 0$. By definition, there must exist p' such that $\delta^2(p', (s', y)) > 0$. As a consequence, by definition of δ^2, there exists a transition $s \overset{y}{\dashrightarrow} s'$ in M and $\varphi((s, x), (s', y)) \neq \{0\}$.

(b) If $\varphi((s, x)(s', y)) =]0, 1]$, then there exists a transition $s \overset{y}{\rightarrow} s'$ in M. As a consequence, by $\mathcal{R}$, there exists $s'_I \in S_I$ such that $s_I \overset{y}{\rightarrow} s'_I$ in I and $s'_I \mathcal{R}\, s'$. Thus $\varphi_I((s_I, x), (s'_I, y)) =]0, 1]$. By definition of δ^1, we know that

$\sum_{p' \in Q^T} \delta^1(p', (s'_I, y)) > 0$, thus there exists $p' \in Q_T$ such that $\delta^1(p', (s'_I, y)) > 0$. Since $s'_I \mathcal{R} s'$ and $s \xrightarrow{y} s'$, we have $\delta^2(p', (s', y)) > 0$, thus

$$\sum_{p'' \in Q^T} \delta^2(p'', (s', y)) > 0.$$

Finally, if $\delta^2(p', (s', y)) > 0$, there exists $s'_I \in S_I$ such that $s'_I \mathcal{R} s'$ and $\delta^1(p', (s'_I, y)) > 0$. By definition of δ^1, we have $p' \mathcal{R}_1(s'_I, y)$. As a consequence, $p' \mathcal{R}_2(s', y)$.

$\mathcal{R}_2$ satisfies the axioms of a satisfaction relation for IMCs, thus $T \in [\![\widehat{M}]\!]$ and finally $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$.

□

Lemma 5. *Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. We have $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!] \Rightarrow I \models M$.*

Proof. Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. Let $\widehat{M} = \langle Q, q_0, A \cup \{\epsilon\}, \varphi, V \rangle$ and $\widehat{I} = \langle Q_I, q_0^I, A \cup \{\epsilon\}, \varphi_I, V_I \rangle$ be the IMCs defined as above.

Suppose that $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$. We prove that $I \models M$.

Let $T = \langle Q_T, p_0, \pi^T, V_T, A \rangle$ be an MC such that $T \in [\![\widehat{I}]\!]$. As a consequence, there exists two satisfaction relations for IMCs $\mathcal{R}_1 \subseteq Q_T \times Q_I$ and $\mathcal{R}_2 \subseteq Q_T \times Q$ such that $p_0 \mathcal{R}_1(s_0^I, \epsilon)$ and $p_0 \mathcal{R}_2(s_0, \epsilon)$. Define the new relation $\mathcal{R} \subseteq S_I \times S$ such that $s_I \mathcal{R} s$ iff there exists $p \in Q_T$ and $x \in (\{\epsilon\} \cup A)$ such that $p \mathcal{R}_1(s_I, x)$ and $p \mathcal{R}_2(s, x)$. We have

1. $p_0 \mathcal{R}_1(s_0^I, \epsilon)$ and $p_0 \mathcal{R}_2(s_0, \epsilon)$. As a consequence, $s_0^I \mathcal{R} s_0$.
2. Let s_I, s, p, x such that $p \mathcal{R}_1(s_I, x)$ and $p \mathcal{R}_2(s, x)$ and let $\delta^1 \in \mathsf{Distr}(Q_T \times Q_I)$ and $\delta^2 \in \mathsf{Distr}(Q_T \times Q)$ be the associated probability distributions.

   (a) Let $y \in A$ and $s'_I \in S_I$ such that $s_I \xrightarrow{y} s'_I$ in I. We prove that there exists $s' \in S$ such that $s \overset{y}{\dashrightarrow} s'$ and $s'_I \mathcal{R} s'$.

   By definition of $\widehat{I}$, we have $\varphi_I((s_I, x), (s'_I, y)) =]0, 1]$. As a consequence, $\sum_{p'' \in Q_T} \delta^1(p'', (s'_I, y)) > 0$. Thus there exists p' in Q_T such that $\delta^1(p', (s'_I, y)) > 0$. By definition of δ^1, we have $p' \mathcal{R}_1(s'_I, y)$, thus $V_T(p') = V_I((s'_I, y)) = \{y\}$.

   Moreover, by definition of δ^1, we have $\sum_{(s''_I, z) \in Q_I} \delta^1(p', (s''_I, z)) = \pi^T(p)(p')$. Since $\delta^1(p', (s'_I, y)) > 0$, we have $\pi^T(p)(p') > 0$.

   By definition of δ^2, we know that $\sum_{(s'', z) \in Q} \delta^2(p', (s'', z)) = \pi^T(p)(p') > 0$. As a consequence, there exists $(s', z) \in Q$ such that $\delta^2(p', (s', z)) > 0$. By definition of δ^2,we have $p' \mathcal{R}_2(s', z)$ and since $V_T(p') = \{y\}$, we must have $z = y$.

   Consequently, $\sum_{p'' \in Q_T} \delta^2(p'', (s', y)) > 0$. By definition of δ^2, we know that $\sum_{p'' \in Q_T} \delta^2(p'', (s', y)) \in \varphi((s, x), (s', y))$, thus $\varphi((s, x), (s', y)) \neq \{0\}$, which

means, by definition of $\widehat{M}$, that there exists a transition $s \overset{y}{\dashrightarrow} s'$ in M. Moreover, there exits $p' \in Q_T$ such that both $p' \mathcal{R}_1 (s'_I, y)$ and $p' \mathcal{R}_2 (s', y)$, thus $s'_I \mathcal{R} s'$.

(b) Let $y \in A$ and $s' \in S$ such that $s \overset{y}{\rightarrow} s'$ in M. We prove that there exists $s'_I \in S_I$ such that $s_I \overset{y}{\rightarrow} s'_I$ in I and $s'_I \mathcal{R} s'$.
By definition of $\widehat{M}$, we have $\varphi((s,x),(s',y)) =]0,1]$. As a consequence, $\sum_{p'' \in Q_T} \delta^2(p'', (s', y)) > 0$. Thus there exists p' in Q_T such that $\delta^2(p', (s', y)) > 0$. By definition of δ^2, we have $p' \mathcal{R}_2 (s', y)$, thus $V_T(p') = V((s', y)) = \{y\}$.
Moreover, by definition of δ^2, we have $\sum_{(s'',z) \in Q} \delta^2(p', (s'', z)) = \pi^T(p)(p')$. Since $\delta^2(p', (s', y)) > 0$, we have $\pi^T(p)(p') > 0$.
By definition of δ^1, we know that $\sum_{(s''_I,z) \in Q_I} \delta^1(p', (s''_I, z)) = \pi^T(p)(p') > 0$. As a consequence, there exists $(s'_I, z) \in Q_I$ such that $\delta^1(p', (s'_I, z)) > 0$. By definition of δ^1, we have $p' \mathcal{R}_1 (s'_I, z)$ and since $V_T(p') = \{y\}$, we must have $z = y$.
Consequently, $\sum_{p'' \in Q_T} \delta^1(p'', (s'_I, y)) > 0$. By definition of δ^1,we know that $\sum_{p'' \in Q_T} \delta^1(p'', (s'_I, y)) \in \varphi_I((s_I, x), (s'_I, y))$, thus $\varphi_I((s,x),(s',y)) \neq \{0\}$, which means, by definition of $\widehat{I}$, that there exists a transition $s_I \overset{y}{\rightarrow} s'_I$ in I (remember that I is a classical transition system). Moreover, there exits $p' \in Q_T$ such that both $p' \mathcal{R}_1 (s'_I, y)$ and $p' \mathcal{R}_2 (s', y)$, thus $s'_I \mathcal{R} s'$.
Finally, $\mathcal{R}$ is a satisfaction relation for MTS, and $I \models M$

□

From the two lemmas stated above, we can infer the following theorem:

Theorem 6. *Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. We have $I \models M \iff [\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$.*

We now define a construction f that builds, for all implementations C of $\widehat{M}$, a corresponding implementation $f(C)$ of M:

Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be a MTS. Let $\widehat{M} = \langle S \times (\{\epsilon\} \cup A), (s_0, \epsilon), \{\epsilon\} \cup A, \varphi, V \rangle$ be the transformation of M defined as above. Let $C = \langle Q, q_0, A, \pi, V' \rangle$ be a MC such that $C \models \widehat{M}$ for some satisfaction relation on IMCs $\mathcal{R}$. Define $f(C) = (Q, q_0, A, \rightarrow)$ the Transition System such that $q \overset{a}{\rightarrow} q'$ whenever $\pi(q, q') > 0$ and $V'(q') = \{a\}$. By construction, it is trivial that (1) $f(C) \models M$ for some satisfaction relation on MTS $\mathcal{R}'$ and (2) $C \models \widehat{f(C)}$ for some satisfaction relation on IMCs $\mathcal{R}''$. These satisfaction relations are defined as follows:

- $q \mathcal{R}' s$ whenever there exists $x \in \{\epsilon\} \cup A$ such that $q \mathcal{R} (s, x)$;
- $q \mathcal{R}'' (q', x)$ whenever $q = q'$.

We now switch to the main theorem, showing that the transformation $M \rightarrow \widehat{M}$ indeed preserves thorough refinement.

Theorem 7. *Let M and M' be two Modal Transition Systems and $\widehat{M}$ and $\widehat{M'}$ be the corresponding IMCs defined as above. We have $M \leq_{\mathsf{T}} M' \iff \widehat{M} \leq_{\mathsf{T}} \widehat{M'}$.*

Proof. Let M and M' be two MTS, and $\widehat{M}$ and $\widehat{M'}$ the corresponding IMCs.

- $\Rightarrow$ Suppose that $M \preceq_T M'$, and let C be a MC such that $C \models \widehat{M}$. We have by construction $f(C) \models M$, thus $f(C) \models M'$. By Theorem 6, we have $[\![\widehat{f(C)}]\!] \subseteq [\![\widehat{M'}]\!]$, and we know that $C \models \widehat{f(C)}$. As a consequence, $C \models \widehat{M'}$.
- $\Leftarrow$ Suppose that $\widehat{M} \preceq_T \widehat{M'}$, and let I be a TS such that $I \models M$. By Theorem 6, we have $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$, thus by hypothesis $[\![\widehat{I}]\!] \subseteq [\![\widehat{M'}]\!]$. Finaly, by Theorem 6, we obtain that $I \models M'$.

□

Crucially, this translation is polynomial. Thus if we had a subexponential algorithm for TR of IMCs, we could use it to obtain a subexponential algorithm for TR of MTSs, which is impossible [43].

6 Determinism

Humans naturally build deterministic models to represent deterministic implementations. Thus deterministic objects form an important class of specifications. It is also known that for other specification langages, determinism allows more efficient reasoning procedures.

In our specification formalism, deciding weak refinement is easier than deciding thorough refinement even though both are in EXPTIME. Nevertheless, since these two refinements do not coincide, in general, a procedure to check weak refinement cannot be used to decide thorough refinement.

Observe that weak refinement has a syntactic definition very much like simulation for transition systems. On the other hand, thorough refinement is a semantic concept, just as trace inclusion for transition systems. It is well known that simulation and trace inclusion coincide for deterministic automata. Similarly, for MTSs it is known that TR coincides with modal refinement for deterministic objects. It is thus natural to define deterministic IMCs and check whether thorough and weak refinements coincide on these objects.

In our context, an IMC is deterministic if, from a given state, one cannot reach two states that share common atomic propositions.

Definition 7 (Determinism)**.** *An IMC $I = \langle Q, q_0, \varphi, A, V\rangle$ is* deterministic *iff for all states $q, r, s \in Q$, if there exists a distribution $\sigma \in \varphi(q)$ such that $\sigma(r) > 0$ and $\sigma(s) > 0$, then $V(r) \neq V(s)$.*

Weak determinism ensures that two states reachable *with the same admissible distribution* always have different valuations. In a semantic interpretation this means that

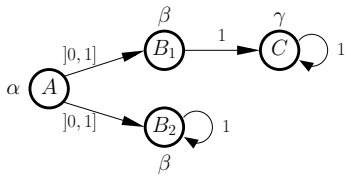


Figure 7: An IMC I whose semantics cannot be captured by a deterministic IMC

there exists no implementation of I, in which two states with the same valuation can be successors of the same source state.

One can also propose another, more syntactic definition of determinism:

Definition 8 (Strong Determinism)**.** *Let $I = \langle Q, q_0, \varphi, A, V\rangle$ be an IMC. I is* strongly deterministic *iff for all states $q, r, s \in Q$, if there exist a probability distribution $\sigma \in \varphi(q)$ such that $\sigma(r) > 0$ and a probability distribution $\rho \in \varphi(q)$ such that $\rho(s) > 0$, then $V(r) \neq V(s)$.*

Strong determinism differs from the notion of determinism presented in Def. 7 in that it requires that, from a given state q, one cannot possibly reach two states r and s with the same set of propositions, even using *two different distributions* (implementations). Checking weak determinism requires solving a cubic number of linear constraints: for each state check the linear constraint of the definition—one per each pair of successors of a state. Checking strong determinism can be done by solving only a quadratic number of linear constraints—one per each successor of each state.

Luckily, due to the convexity of the set of admissible distributions in a state, these two notions coincide for IMCs, so the more efficient, strong determinism can be used in algorithms:

Theorem 8. *An IMC I is deterministic iff it is strongly deterministic.*

Proof. It directly follows from the definitions that strong determinism implies weak determinism. We prove that if an IMC I is not strongly deterministic, then it is not weakly deterministic either.

Let $I = \langle Q, q_0, \varphi, A, V\rangle$ be an IMC. If I is not strongly deterministic, then there exist two admissible distributions on next states for q: σ and $\rho \in \varphi(q)$ such that $\sigma(r) > 0$, $\sigma(s) = 0$, $\rho(r) = 0$, $\rho(s) > 0$ and $V(r) = V(s)$. In order to prove that I is not weakly deterministic, we build a distribution γ that we prove correct with respect to the interval specifications, i.e. $\gamma \in \varphi(q)$, and such that $\gamma(r) > 0$ and $\gamma(s) > 0$.

Since $\sigma(r) > 0$, there exists $a > 0$ such that $\varphi(q)(r) = [0, a]$ or $[0, a[$. Moreover, since $\rho(s) > 0$, there exists $b > 0$ such that $\varphi(q)(s) = [0, b]$ or $[0, b[$. Let $c = \mathsf{Min}(a, b)$, and define $\gamma(q') = \sigma(q')$ for all $q' \notin \{r, s\}$, $\gamma(r) = \sigma(r) - c/2$, and $\gamma(s) = c/2$. By construction, $\gamma \in \varphi(q)$ and we have $\gamma(r) > 0$ and $\gamma(s) > 0$. As a consequence, I is not weakly deterministic. Finally, an IMC I is strongly deterministic iff it is also weakly deterministic.

□

It is worth mentioning that deterministic IMCs are a strict subclass of IMCs. Figure 7 shows an IMC I whose set of implementations cannot be represented by a deterministic IMC.

We now state the main theorem of the section that shows that for deterministic IMCs, the weak refinement, and indeed also the strong refinement, correctly capture the thorough refinement:

Theorem 9. *For deterministic IMCs I and I' with no inconsistent states, the following statements are equivalent,*

1. *I thoroughly refines I',*
2. *I weakly refines I', and*
3. *I strongly refines I'.*

Proof. It directly follows the definitions that (3) implies (2) and (2) implies (1). We will prove that (1) implies (2), and then that (2) implies (3).

Let $I_1 = \langle Q^1, q_0^1, \varphi_1, A, V_1\rangle$ and $I_2 = \langle Q^2, q_0^2, \varphi_2, A, V_2\rangle$ be two consistent and deterministic IMCs such that $[\![I_1]\!] \subseteq [\![I_2]\!]$.
First, remark that it is safe to suppose that implementations have the same set of atomic propositions as I_1 and I_2.

1. Let $\mathcal{R} \subseteq Q^1 \times Q^2$ be such that $r \,\mathcal{R}\, s$ iff for all MC C and state p of C, $p \models r \Rightarrow p \models s$. Since we consider pruned IMCs, there exist implementations for all states.

   Consider r and s such that $r \,\mathcal{R}\, s$.

   (a) By definition of $\mathcal{R}$, there exists a MC C and a state p of C such that $p \models r$ and $p \models s$. Thus $V_C(p) = V_1(r)$ and $V_C(p) = V_2(s)$. As a consequence, $V_1(r) = V_2(s)$.

   (b) Consider $\rho \in \varphi_1(r)$ and build the MC $C = \langle Q^1, q_0^1, \pi, A, V_C\rangle$ such that for all $q \in Q^1$,

   - $V_C(q) = V_1(q)$;
   - If $q \neq r$, $\pi(q)$ is any distribution in $\varphi_1(q)$. At least one exists because I_1 is pruned;
   - $\pi(r) = \rho$.

   When necessary, we will address state q of C as q_C to differentiate it from state q of I_1. We will now build the correspondence function δ.

   C clearly satisfies I_1 with a satisfaction relation $\mathcal{R}_1 = \mathsf{Identity}$, and $r_C \models r$. By hypothesis, we thus have $r_C \models s$. Consider $\mathcal{R}_2$ the satisfaction relation such that $r_C \,\mathcal{R}_2\, s$ and δ_2 the corresponding correspondence function. Let $\delta = \delta_2$.

   (c) As a consequence,

i. By construction of δ, we have that for all $q \in Q^1$, $\delta(q)$ is a probability distribution;

ii. By definition of the *s*atisfaction relation $\mathcal{R}_2$, we have that for all $s' \in Q^2$, $\sum_{q_C \in Q^1} \rho(q_C)\delta_2(q_C)(s') \in \varphi_2(s)(s')$. As a consequence, for all $s' \in Q^2$, $\sum_{q \in Q^1} \rho(q)\delta(q)(s') \in \varphi_2(s)(s')$.

2. Let $r' \in Q^1$ and $s' \in Q^2$ be such that $\delta_{r's'} \neq 0$. By definition of C and δ, we have ${r'}_C \models r'$ and ${r'}_C \models s'$. We want to prove that for all implementations C' and state p' in C', $p' \models r'$ implies $p' \models s'$.

   Suppose that this is not the case. There exists an implementation $C' = \langle P, o, \pi', A, V' \rangle$ and a state p' of C' such that $p' \models r'$ and $p' \not\models s'$. Let $\mathcal{R}'$ be the satisfaction relation witnessing $p' \models r'$.

   Consider the MC $\widehat{C} = \langle \widehat{Q^1} \cup \widehat{P}, \widehat{q_0^1}, \widehat{\pi}, A, \widehat{V} \rangle$. Intuitively, $\widehat{Q^1}$ corresponds to C and $\widehat{P}$ corresponds to C'. The state ${r'}_C$ (called $\widehat{r'}$ in $\widehat{C}$) will be the link between the two and its outgoing transitions will be the ones of p'. Define

   - $\widehat{\pi}(\widehat{q_1})(\widehat{q_2}) = \pi(q_1)(q_2)$ if $q_1, q_2 \in Q^1$ and $\widehat{q_1} \neq \widehat{r'}$;
   - $\widehat{\pi}(\widehat{r'})(q_2) = 0$ if $q_2 \in Q^1$;
   - $\widehat{\pi}(\widehat{q_1})(\widehat{p_2}) = 0$ if $q_1 \in Q^1$ and $\widehat{q_1} \neq \widehat{r'}$ and $p_2 \in \widehat{P}$;
   - $\widehat{\pi}(\widehat{r'})(\widehat{p_2}) = \pi'(p')(p_2)$ if $p_2 \in P$;
   - $\widehat{\pi}(\widehat{p_1})(\widehat{q_2}) = 0$ if $p_1 \in P$ and $q_2 \in Q^1$;
   - $\widehat{\pi}(\widehat{p_1})(\widehat{p_2}) = \pi'(p_1)(p_2)$ if $p_1, p_2 \in P$;
   - $\widehat{V}(\widehat{q}) = V_1(q)$ if $q \in Q^1$;
   - $\widehat{V}(\widehat{p_1}) = V'(p_1)$ if $p_1 \in P$.

   We want to prove that $\widehat{r'}$ satisfies s'. This should imply that $p'_{C'}$ also satisfies s', which is absurd.

   Consider the relation $\widehat{\mathcal{R}}$ between the states of $\widehat{C}$ and the states of I_1 defined as follows :

$$\begin{aligned}\widehat{\mathcal{R}} =& \{(\widehat{q^1}, {q^1}') \mid (q^1_C, {q^1}') \in R_1 \text{ and } \widehat{q^1} \neq \widehat{r'}\} \cup \\ & \{(\widehat{p^1}, {q^1}') \mid (p^1, {q^1}') \in \mathcal{R}'\} \cup \\ & \{(\widehat{r'}, {q^1}') \mid p' \,\mathcal{R}'\, {q^1}'\}\end{aligned}$$

   Intuitively, $\widehat{\mathcal{R}}$ is equal to $\mathcal{R}_1$ for the states $\widehat{q^1} \in \widehat{Q^1}$, except $\widehat{r'}$, and equal to $\mathcal{R}'$ for the states $\widehat{p^1} \in \widehat{P}$. The states related to $\widehat{r'}$ are the ones that were related to p' with $\mathcal{R}'$.

We will show that $\widehat{\mathcal{R}}$ is a satisfaction relation between $\widehat{C}$ and I_1.

Let t, w be such that $t\widehat{\mathcal{R}}w$. For all the pairs where $t \neq \widehat{r'}$, the conditions of the satisfaction relation obviously still hold because they held for $\mathcal{R}_1$ if $t \in \widehat{Q^1}$ and for $\mathcal{R}'$ otherwise. It remains to check the conditions for the pairs where $t = \widehat{r'}$.

Consider w such that $\widehat{r'}\widehat{\mathcal{R}}w$.

(a) Since r'_C and $p'_{C'}$ are both implementations of r', it is clear that $\widehat{V}(\widehat{r'}) = \widehat{V}(p')$. As $p' \mathcal{R}' w$, we know that $V'(p') = V_1(w)$. Thus, $\widehat{V}(\widehat{r'}) = V_1(w)$.

(b) Consider the correspondence function $\delta' : P \to (Q^1 \to [0,1])$ given by $p' \mathcal{R}' w$. Let $\widehat{\delta} : (\widehat{Q^1} \cup \widehat{P}) \to (Q^1 \to [0,1])$ be such that $\widehat{\delta}(\widehat{p^1}) = \delta'(p^1)$ whenever $\widehat{p^1} \in \widehat{P}$. Obviously, this is still a probability distribution on Q^1, and it is such that

   i. for all $q^1 \in Q^1$,

$$\sum_{t \in \widehat{Q^1} \cup \widehat{P}} \widehat{\pi}(r')(t)\widehat{\delta}(t)(q^1) = \sum_{\widehat{p_2} \in \widehat{P}} \pi'(p')(p_2)\widehat{\delta}(\widehat{p_2})(q^1)$$
$$= \sum_{p_2 \in P} \pi'(p')(p_2)\delta'(p_2)(q^1).$$

   By definition of δ', this is contained in $\varphi_1(w)(q^1)$.

   ii. Moreover, if $\widehat{\pi}(\widehat{r'})(t) \neq 0$ and $\widehat{\delta}(t)(q^1) \neq 0$, then $t\widehat{\mathcal{R}}q^1$. We only need to consider $t = \widehat{p_1} \in \widehat{P}$ (since otherwise $\widehat{\pi}(\widehat{r'})(t) = 0$) and q^1 such that $\widehat{\delta}(\widehat{p_1})(q^1) \neq 0$. In this case, $\delta'(p_1)(q^1) \neq 0$. As δ' is a witness of $p' \mathcal{R}' w$, it has to be that $p_1 \mathcal{R}' q^1$, which implies, by definition of $\widehat{\mathcal{R}}$, that $t\widehat{\mathcal{R}}q^1$.

Finally, $\widehat{C}$ satisfies I_1, and in particular, $\widehat{r} \models r$. As $r \mathcal{R} s$, it implies that $\widehat{r} \models s$. As a consequence, there exists $\delta'' : (\widehat{Q^1} \cup \widehat{P}) \to (Q^2 \to [0,1])$ such that, for all $q^2 \in Q^2$,

$$\sum_{t \in \widehat{Q^1} \cup \widehat{P}} \widehat{\pi}(\widehat{r})(t)\delta''(t)(q^2) \in \varphi_2(s)(q^2)$$

(A) Consider $q^2 \neq s'$ such that $V_2(q^2) = V_2(s')$. Due to determinism of I_2, and to the fact that s' is accessible from s, we have $\varphi_2(s)(q^2) = \{0\}$. Since $\widehat{\pi}(\widehat{r})(\widehat{r'}) \neq 0$ and $\widehat{\pi}(\widehat{r})(\widehat{r'})\delta''(\widehat{r'})(q^2)$ is part of the sum above, we must have $\delta''(\widehat{r'})(q^2) = 0$.

(B) Consider q^3 such that $V_2(q^3) \neq V_2(s') = V_1(r')$. It is clear that $\delta''(\widehat{r'})(q^3) = 0$ since δ'' is witnessing satisfaction between $\widehat{C}$ and I_2.

(C) Moreover, since $\widehat{\pi}(\widehat{r})(\widehat{r'}) > 0$, we know that $\delta''(\widehat{r'})$ is a probability distribution over Q^2.

According to (A) and (B), the only non-zero value in the distribution in (C) must be $\delta''(\widehat{r'})(s')$. Since δ'' is witnessing $\widehat{C} \models I_2$, this means that $\widehat{r'} \models s'$.

By construction, $\widehat{r'}$ and p' only differ by state names. This contradicts the assumption that $p' \not\models s'$. Thus $r' \mathcal{R} s'$, and $\mathcal{R}$ is a weak refinement relation.

Finally, we have by hypothesis that $[\![I_1]\!] \subseteq [\![I_2]\!]$, which implies that $q_0^1 \mathcal{R} q_0^2$. We thus have (1) implies (2).

□

We now prove that (2) implies (3). The following lemma is a direct consequence of determinism. It states that correspondence functions associated to a satisfaction relation for a deterministic IMC are of a particular form.

Lemma 10. *Let $I = \langle Q, q_0, \varphi, A, V\rangle$ be a deterministic IMC. Let $C = \langle P, p_0, \pi, A, V_C\rangle \in [\![I]\!]$ be a MC and let $\mathcal{R}$ be a satisfaction relation such that $p_0 \mathcal{R} q_0$. Let $p \in P$ and $q \in Q$ be such that $p \mathcal{R} q$, and let δ be the associated correspondence function. We have*

$$\forall p' \in P,\ \pi(p)(p') \neq 0 \Rightarrow |\{q' \in Q \mid \delta(p')(q') \neq 0\}| = 1. \tag{2}$$

Obviously, the same holds for correspondence functions associated to refinement relations between deterministic IMCs.

Let $I_1 = \langle Q^1, q_0^1, \varphi_1, A, V_1\rangle$ and $I_2 = \langle Q^2, q_0^2, \varphi_2, A, V_2\rangle$ be two deterministic IMCs such that $I_1 \preceq_W I_2$ with a weak refinement relation $\mathcal{R}$. We prove that $\mathcal{R}$ is in fact a strong refinement relation.

Let $p \in Q^1$ and $q \in Q^2$ be such that $p \mathcal{R} q$.

1. By hypothesis, $V_1(p) = V_2(q)$;
2. We know that for all probability distribution $\sigma \in \varphi_1(p)$, there exists a correspondence function δ^σ satisfying the axioms of a (weak) refinement relation. We will build a correspondence function δ^0 that will work for all σ. Let $p' \in Q^1$.
   - If for all $\sigma \in \varphi_1(p)$, we have $\sigma(p') = 0$, then let $\delta^0(p', q') = 0$ for all $q' \in Q^2$;
   - Else, consider $\sigma \in \varphi_1(p)$ such that $\sigma(p') \neq 0$. By hypothesis, there exists a correspondence function δ^σ associated to $p \mathcal{R} q$. Let $\delta^0(p') = \delta^\sigma(p')$. By Lemma 10, there is a single $q' \in Q^2$ such that $\delta^\sigma(p')(q') \neq 0$. Moreover, by definition of δ^σ, we know that $\sum_{q'' \in Q^2} \delta^\sigma(p')(q'') = 1$, thus $\delta^\sigma(p')(q') = 1$.
     Suppose there exists $\rho \neq \sigma \in \varphi_1(p)$ such that $\rho(p') \neq 0$. Let δ^ρ be the associated correspondence function. As for σ, there exists a unique $q'' \in Q^2$ such that $\delta^\rho(p')(q'') \neq 0$. Moreover $\delta^\rho(p')(q'') = 1$. By definition of δ^σ and δ^ρ, we have

$$\mu : q''' \mapsto \sum_{p'' \in Q^1} (\sigma(p'')\delta^\sigma(p'')(q''')) \in \varphi_2(q)$$
$$\nu : q''' \mapsto \sum_{p'' \in Q^1} (\rho(p'')\delta^\rho(p'')(q''')) \in \varphi_2(q)$$

Moreover, both $\mu(q') > 0$ and $\nu(q'') > 0$. By determinism of I_2, this implies $q' = q''$.

As a consequence, we have $\delta^\sigma(p') = \delta^\rho(p')$, so $\forall \gamma \in \varphi_1(p)$, if $\gamma(p') > 0$, then $\delta^\gamma(p') = \delta^0(p')$.

Finally, consider δ^0 defined as above. Let $\sigma \in \varphi_1(p)$. We have

(a) if $\sigma(p') > 0$, then $\delta^0(p') = \delta^\sigma(p')$ is a distribution over Q^2;

(b) for all $q' \in Q^2$,

$$\sum_{p' \in Q^1} (\sigma(p')\delta^0(p')(q')) = \sum_{p' \in Q^1} (\sigma(p')\delta^\sigma(p')(q')) \in \varphi_2(q)(q') \text{ by definition of } \delta^\sigma;$$

(c) if $\delta^0(p')(q') > 0$, then there exists $\sigma \in \varphi_1(p)$ such that $\delta^0(p')(q') = \delta^\sigma(p'q') > 0$, thus $p' \mathcal{R} q'$ by definition of δ^σ.

Finally, $\mathcal{R}$ is a strong refinement relation.

□

7 Common Implementation and Consistency

We now turn our attention to the problem of implementation of several IMC specifications by the same probabilistic system modeled as a Markov Chain. We start with defining the problem:

Definition 9 (Common Implementation (CI))**.** *Given $k > 1$ IMCs I_i,$i = 1 \dots k$, does there exist a Markov Chain C such that $C \models I_i$ for all i?*

Somewhat surprisingly we find out that, similarly to the case of TR, the CI problem is not harder for IMCs than for modal transition systems:

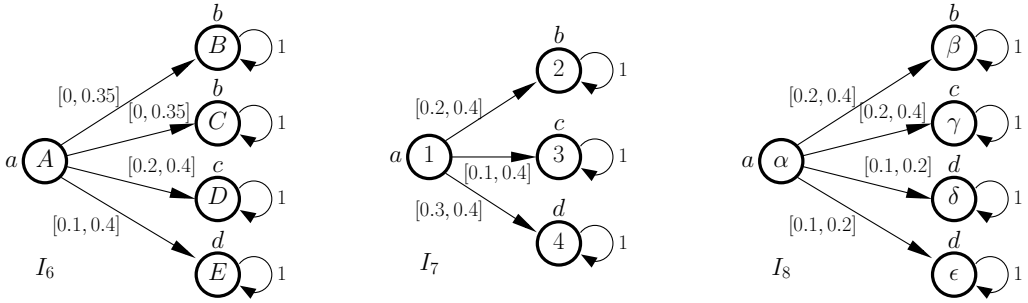
Theorem 11. *Deciding the existence of a CI between k IMCs is EXPTIME-complete in general.*

Lower Bound. To establish a lower bound for common implementation, we propose a reduction from the common implementation problem for modal transition systems (MTS). This latter problem has recently been shown to be EXPTIME-complete when the number of MTS is not known in advance and PTIME-complete otherwise [50]. We first propose the following theorem.

Theorem 12. *Let M_i be MTSs for $i = 1, \dots, k$. We have*

$$\exists I \forall i : I \models M_i \iff \exists C \forall i : C \models \widehat{M_i},$$

where I is a transition system, C is a Markov Chain and $\widehat{M_i}$ is the IMC obtained with the transformation defined in Section 5.



(a) I_6 **(b)** I_7 **(c)** I_8

Figure 8: IMCs I_6, I_7, and I_8

Proof. $\Rightarrow$: This direction can be proven by showing that for arbitrary $j \in \{1, \ldots, k\}$, $[\![\widehat{I}]\!] \subseteq [\![\widehat{M_j}]\!]$. This is indeed the result of Theorem 6. Now pick a $C \in [\![\widehat{I}]\!]$, and the result follows.
$\Leftarrow$: Assume that there exists a C such that $C \models \widehat{M_i}$ for all $i = 1, \ldots, k$. With the transformation defined in section 5, an implementation I for all M_i for all i can be constructed as $f(C)$. □

Upper Bound. To address the upper bound we first propose a simple construction to check if there exists a CI for two IMCs. We start with the definition of *consistency relation* that witnesses a common implementation between two IMCs.

Definition 10. *Let* $I_1 = \langle Q_1, q_0^1, \varphi_1, A, V_1 \rangle$ *and* $I_2 = \langle Q_2, q_0^2, \varphi_2, A, V_2 \rangle$ *be IMCs. Then* $\mathcal{R} \subseteq Q_1 \times Q_2$ *is a* consistency relation *on the states of* I_1 *and* I_2 *iff whenever* $(u, v) \in \mathcal{R}$ *then*

- $V_1(u) = V_2(v)$,
- *there exists a* $\rho \in \mathsf{Distr}(Q_1 \times Q_2)$ *such that*
  1. $\forall u' \in Q_1 : \sum_{v' \in Q_2} \rho(u', v') \in \varphi_1(u)(u') \ \wedge \ \forall v' \in Q_2 : \sum_{u' \in Q_1} \rho(u', v') \in \varphi_2(v)(v')$, *and*
  2. $\forall (u', v') \in Q_1 \times Q_2$ *st.* $\rho(u', v') > 0$, *then* $(u', v') \in \mathcal{R}$.

We illustrate the definition of a consistency relation in the following example.

Example 2. *Consider the three IMCs in Figure 8. We construct a consistency relation* $\mathcal{R}$ *for* $k = 3$. *The triple* $(A, 1, \alpha)$ *is in the relation* $\mathcal{R}$ *witnessed by the distribution* ρ *that assigns* $\frac{1}{6}$ *to* $(B, 2, \beta)$, $\frac{1}{6}$ *to* $(C, 2, \beta)$, $\frac{1}{3}$ *to* $(D, 3, \gamma)$, $\frac{1}{6}$ *to* $(E, 4, \delta)$, *and* $\frac{1}{6}$ *to* $(E, 4, \epsilon)$. *The triples that are given positive probability by* ρ *are also in the relation each by the distribution assigning probability 1 to itself. A common implementation* $C = \langle P, p_0, \pi, A, V_C \rangle$ *can be constructed as follows:* $P = \{q | q \in \mathcal{R}\}$, $p_0 = (A, 1, \alpha)$, $V_C(p)$ *is inherited from* I_6, I_7, *and* I_8, *and* $\pi(p)(p') = \rho(p')$, *where* ρ *is the distribution witnessing that* $p \in \mathcal{R}$.

We now prove that the existence of a consistency relation is equivalent to the existence of a common implementation, in the case of $k = 2$. The above definition and the following theorem extends to general k.

Theorem 13. *Let $I_1 = \langle Q_1, q_0^1, \varphi_1, A, V_1\rangle$ and $I_2 = \langle Q_2, q_0^2, \varphi_2, A, V_2\rangle$ be IMCs. I_1 and I_2 have a common implementation iff there exists a consistency relation $\mathcal{R}$ such that $q_0^1 \,\mathcal{R}\, q_0^2$.*

Proof. $\Rightarrow$: Assume that there exists a MC $C = \langle P, p_0, \pi, A, V_C\rangle$ such that $C \models I_1$ and $C \models I_2$. This implies that there exists satisfaction relations $\mathcal{R}_1 \subseteq P \times Q_1$ and $\mathcal{R}_2 \subseteq P \times Q_2$ such that $p_0 \,\mathcal{R}_1\, q_0^1$ and $p_0 \,\mathcal{R}_2\, q_0^2$.

A relation $\mathcal{R}$ is constructed as $\{(q_1, q_2) | \exists p \in P : p \,\mathcal{R}_1\, q_1 \ \wedge\ p \,\mathcal{R}_2\ q_2\}$. We now prove that $\mathcal{R}$ is a consistency relation relating q_0^1 and q_0^2; indeed $(q_0^1, q_0^2) \in \mathcal{R}$ because $p_0 \,\mathcal{R}_1\, q_0^1$ and $p_0 \,\mathcal{R}_2\, q_0^2$. Let $(q_1, q_2) \in \mathcal{R}$ and $p \in P$ be such that $p \,\mathcal{R}_1\, q_1$ and $p \,\mathcal{R}_2\, q_2$.

1. By $\mathcal{R}_1$ and $\mathcal{R}_2$, $V_1(q_1) = V_C(p) = V_2(q_2)$
2. Let δ_1 and δ_2 be the correspondence functions witnessing $p \,\mathcal{R}_1\, q_1$ and $p \,\mathcal{R}_2\, q_2$, and let $\rho \in \mathsf{Distr}(Q_1 \times Q_2)$ be such that

$$\rho(q_1', q_2') = \sum_{p' \in P \text{ st. } \pi(p)(p') > 0} \pi(p)(p')\delta_1(p', q_1')\delta_2(p', q_2'). \tag{3}$$

   Since $\sum_{q_1' \in Q_1} \sum_{q_2' \in Q_2} \rho(q_1', q_2') = 1$, ρ is indeed a distribution on $Q_1 \times Q_2$.

   Let $u' \in Q_1$.

$$\begin{aligned}
\sum_{v' \in Q_2} \rho(u', v') &= \sum_{(v' \in Q_2)\ (p' \in P \text{ st. } \pi(p)(p') > 0)} \pi(p)(p')\delta_1(p', u')\delta_2(p', v') \\
&= \sum_{p' \in P \text{ st. } \pi(p)(p') > 0} \pi(p)(p')\delta_1(p', u') \sum_{v' \in Q_2} \delta_2(p', v') \\
&= \sum_{p' \in P \text{ st. } \pi(p)(p') > 0} \pi(p)(p')\delta_1(p', u') \quad \text{by definition of } \delta_2 \\
&\in \varphi_1(q_1)(u') \quad \text{by definition of } \delta_1.
\end{aligned}$$

   Similarly, for all $v' \in Q_2$, $\sum_{u' \in Q_1} \rho(u', v') \in \varphi_2(v)(v')$.

3. Let $q_1' \in Q_1$ and $q_2' \in Q_2$ be states such that $\rho(q_1', q_2') > 0$. Then at least one term in Eq. (3) is positive. Thus, there exists p' such that

$$\pi(p)(p')\delta_1(p', q_1')\delta_2(p', q_2') > 0.$$

   This implies that all factors are positive, and by definition of δ_1 and δ_2, we have that $(p', q_1') \in \mathcal{R}_1$ and $(p', q_2') \in \mathcal{R}_2$ and therefore $q_1' \,\mathcal{R}\, q_2'$.

This proves that $\mathcal{R}$ is a consistency relation.

$\Leftarrow$: Assume that there exists a consistency relation $\mathcal{R}$ relating q_0^1 and q_0^2. We now construct a common implementation C, such that $C \models I_1$ and $C \models I_2$; we prove the former first. Let $C = \langle P, p_0, \pi, A, V_C\rangle$ be such that

- $P = \{(q_1, q_2) \in Q_1 \times Q_2 \mid q_1 \mathcal{R} q_2\}$
- $p_0 = (q_0^1, q_0^2)$
- $V_C((q_1, q_2)) = V_1(q_1) = V_2(q_2)$ by definition of $\mathcal{R}$
- For each $(q_1, q_2), (q_1', q_2') \in P$, $\pi((q_1, q_2)(q_1', q_2')) = \rho(q_1', q_2')$, where ρ is the distribution witnessing the membership of (q_1, q_2) in $\mathcal{R}$.

To show satisfaction between C and I_1, the relation $\mathcal{R}_s$ is used. It is defined as follows: for all $(u, v) \in P$, $(u, v) \mathcal{R}_s w$ iff $u = w$. We now show that $\mathcal{R}_s$ is a satisfaction relation between C and I_1.
Let $(u, v) \in P$ be such that $(u, v) \mathcal{R}_s u$.

1. By definition of C, $V_C(u, v) = V_1(u)$
2. Let δ be the correspondence function such that: $\delta((u', v'), q_1) = 1$ if $u' = q_1$ and 0 else.
   (a) Let $(u', v') \in P$ be such that $\pi(u, v)(u', v') > 0$. $\delta((u', v'))$ is a distribution by definition.
   (b) Let $q_1 \in Q_1$.

$$\begin{aligned}\sum_{(u',v') \in P} \pi(u, v)(u', v')\delta((u', v'), q_1) &= \sum_{(q_1,v') \in P} \pi((u, v), (q_1, v')) \\ &= \sum_{v' \in Q_2} \rho(q_1, v') \\ &\in \varphi_1(u)(q_1) \quad \text{by definition of } \mathcal{R}\,.\end{aligned}$$

   (c) Let $(u', v') \in P$ and $q_1 \in Q_1$ be such that $\delta((u', v'), q_1) > 0$. Then $u' = q_1$ and by definition, $(u', v') \mathcal{R}_s q_1$.

Consequently, $\mathcal{R}_s$ is a satisfaction relation, and thus $C \models I_1$. Analogously, it can be shown that $C \models I_2$. Finally C is a common implementation of I_1 and I_2. □

As a consequence, deciding the existence of a common implementation between 2 IMCs is PTIME-complete. For the general problem of common implementation of k IMCs, we can extend the above definition of consistency relation to the k-ary relation in the obvious way, and the algorithm becomes exponential in the number of IMCs k, as the size of the state space $\prod_{i=1}^{k} |Q_i|$ is exponential in k.

As a side effect we observe that, exactly like MTSs, CI becomes polynomial for any constant value of k, i.e. when the number of components to be checked is bounded by a constant.

Consistency A related problem is the one of checking consistency of a single IMC I, i.e. whether there exists a Markov chain M such that $M \models I$.

Definition 11 (Consistency (C))**.** *Given an IMC I, does it hold that $[\![I]\!] \neq \emptyset$?*

It turns out that, in the complexity theoretic sense, this problem is easy:

Theorem 14. *The problem C, to decide if a single IMC is consistent, is polynomial time solveable.*

Proof. Given an IMC $I = \langle Q, q_0, \varphi, A, V \rangle$, this problem can be solved by constructing a consistency relation over $Q \times Q$ (as if searching for a common implementation of Q with itself). Now there exists an implementation of I iff there exists a consistency relation containing (q_0, q_0). Obviously, this can be checked in polynomial time. □

The fact that C can be decided in polynomial time casts an interesting light on the ability of IMCs to express inconsistency. On one hand, one can clearly specify inconsistent states in IMCs (simply by giving intervals for successor probabilities that cannot be satisfied by any distribution). On the other hand, this inconsistency appears to be local. It does not induce any global constraints on implementations; it does not affect consistency of other states. In this sense IMCs are weaker than *mixed transition systems* [75]. Mixed transition systems relax the requirement of modal transition systems, not requiring that $(\rightarrow) \subseteq (\dashrightarrow)$. It is known that C is trivial for modal transition systems, but EXPTIME-complete for mixed transition systems [50]. Clearly, with a polynomial time C, IMCs cannot possibly express global behaviour inconsistencies in the style of mixed transition systems, where the problem is much harder.

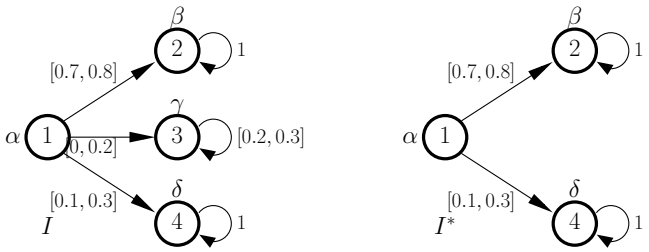
We conclude the section by observing that, given the IMC I and a consistency relation $\mathcal{R} \subseteq Q \times Q$, it is possible to derive a *pruned* IMC $I^* = \langle Q^*, q_0^*, \varphi^*, A, V^* \rangle$ that contains no inconsistent states and accepts the same set of implementations as I.

The construction of I^* is as follows: $Q^* = \{q \in Q | (q, q) \in \mathcal{R}\}$, $q_0^* = q_0$, $V^*(q^*) = V(q^*)$ for all $q^* \in Q^*$, and for all $q_1^*, q_2^* \in Q^*$, $\varphi^*(q_1^*)(q_2^*) = \varphi(q_1^*)(q_2^*)$.

Theorem 15. *Consider an IMC I and its pruned IMC I^*. It holds that $[\![I]\!] = [\![I^*]\!]$.*

Proof. 1. We first prove that $[\![I]\!] \subseteq [\![I^*]\!]$. Let $\mathcal{R} \subseteq Q \times Q$ be a consistency relation such that $(q_0, q_0) \in \mathcal{R}$, and let $C = \langle P, p_0, \pi, A, V_C \rangle$ be a MC such that $C \models I$ with satisfaction relation $\mathcal{R}_s$. We build a satisfaction relation $\mathcal{R}'_s \subseteq P \times Q^*$ where $p \,\mathcal{R}' \, q^*$ iff there exists $q \in Q$ such that $p \,\mathcal{R}_s \, q$ and $q = q^*$. Let $p \in P$, $q \in Q$, and $q^* \in Q^*$ be such that $(p, q^*) \in \mathcal{R}'$. We now show that $\mathcal{R}'$ is a satisfaction relation between P and I^*.

- By construction, $V_C(p) = V^*(q^*)$.
- Let $\delta_1 \in \mathsf{Distr}(P \times Q)$ be the distribution witnessing $p \,\mathcal{R}_s \, q$. The distribution $\delta_2 \in \mathsf{Distr}(P \times Q^*)$ is chosen identical to δ_1. We know that for all $q' \in Q$ such that $\neg\exists \sigma \in \varphi(q')$ then for all $p' \in P$, we have that $\delta_1(p', q') = 0$. To see this, assume the contrary, namely that $\delta_1(p', q') \neq 0$ for a $p' \in P$ and a



(a) IMC I **(b)** Pruned IMC I^*

Figure 9: An IMC and its pruned version

$q' \in Q$ for which $\neg\exists\sigma \in \varphi(q')$; then $p' \mathcal{R}_s q'$. By the definition af satisfaction, q' allows a distribution, which is a contradiction.

Since δ_1 satisfies the axioms of satisfaction, then δ_2 also satisfies them.

2. To show that $[\![I^*]\!] \subseteq [\![I]\!]$, we use the same reasoning as above.

By mutual inclusion, $[\![I]\!] = [\![I^*]\!]$. □

An illustration of pruning is given in the following example.

Example 3. *Consider the IMC I in Figure 9a. Building a consistency relation, we see that* $(1,1)$ *is in the relation witnessed by the distribution assigning probability* 0.8 *to* $(2,2)$ *and* 0.2 *to* $(4,4)$*. This probability distribution "avoids" the inconsistent state* $(3,3)$*; this state does not admit a probability distribution. Likewise,* $(2,2)$ *and* $(3,3)$ *are in the relation, witnessed by the distributions that gives probability 1 to* $(2,2)$ *and* $(3,3)$*, respectively. I^* is shown in Figure 9b.*

8 Conclusion and Future Work

This paper provides new results for IMCs [46, 76, 77, 78] that is a specification formalism for probabilistic systems. We have studied the expressiveness and complexity of three refinement preorders for IMCs. The results are of interest as existing articles on IMCs often use one of these preorders to compare specifications (for abstractions) [46, 47, 48]. We have established complexity bounds and decision procedures for these relations, first introduced in [46]. Finally, we have studied the common implementation problem that is to decide whether there exists an implementation that can match the requirements made by two or more specifications. Our solution is constructive in the sense that it can build such a common implementation.

Our results are robust with respect to simple variations of IMCs. For example sets of sets of propositions can be used to label states, instead of sets of propositions. This extends the power of the modeling formalism, which now can not only express abstractions over probability distributions, but also over possible state valuations. Similarly,

an initial distribution, or even an interval constraint on the initial distribution, could be used instead of the initial state in IMCs without affecting the results.

In the future we expect to see whether our complexity results can be extended to CMCs [63]—an already mentioned generalization of IMCs, which enjoys good closure properties. Furthermore, in order to improve efficiency of tools, it would be desirable to investigate whether IMCs could be used as an abstraction in counter-example guided abstraction-refinement [79] decision procedures for CMCs.

In [47, 49], Katoen et al. have proposed an extension of IMCs to the continuous timed setting. It would be interesting to see whether our results extend to this new model. Another interesting future work would be to extend our results to other specification formalisms for systems that mix both stochastic and non-deterministic aspects. Among them, one finds probabilistic automata [80] where weak/strong refinement would be replaced by probabilistic simulation [6].

Markov set-chains allow iterative approximation of implementations with increasing state space size. It would be interesting to investigate if these could be used to define size-parameterized versions of our decision problems, and whether these could be solved by iterative approximations.

Paper B – Constraint Markov Chains

Benoit Caillaud
INRIA/IRISA, Rennes

Benoit Delahaye
INRIA/IRISA, Rennes

Kim G. Larsen
Aalborg University

Axel Legay
INRIA/IRISA, Rennes

Mikkel L. Pedersen
Aalborg University

Andrzej Wasowski
IT University, Copenhagen

1 Abstract

Notions of specification, implementation, satisfaction, and refinement, together with operators supporting stepwise design, constitute a specification theory. We construct such a theory for Markov Chains (MCs) employing a new abstraction of a Constraint MC. Constraint MCs permit rich constraints on probability distributions and thus generalize prior abstractions such as Interval MCs. Linear (polynomial) constraints suffice for closure under conjunction (respectively parallel composition). This is the first specification theory for MCs with such closure properties. We discuss its relation to simpler operators for known languages such as probabilistic process algebra. Despite the generality, all operators and relations are computable.

2 Introduction

Compositional design [1] is a research field, which aims at development of mathematical foundations for reasoning about components. Usually this is achieved by specifying and analyzing the interfaces of components in order to infer global properties of a system in an incremental way. One popular approach in this area is the work on type systems, and in particular, on type systems for modules in the programming language community. Another approach, in the verification area, is the work on *specification theories*, which provide a modeling language for designing, evolving and advisedly reusing components with formal guarantees. For example, a large system, or a complex communication

protocol, can be designed *step-wise*—by building more and more refined models—and analyzed *piece-wise* by analyzing fragments of models and reasoning about the properties of the parallelcomposition. The hope is that this way we can combat the size and complexity of modern systems.

For functional analysis of discrete-time non-probabilistic systems, the theory of Modal Transition Systems (MTSs) [42, 81] provides a specification formalism supporting refinement as well as conjunction and parallel composition. It has been recently applied to construct interface theories [82, 83], which are extensions of classical interface automata proposed by de Alfaro et al. [84, 85, 86, 87, 88].

As soon as systems include randomized algorithms, probabilistic protocols, or interact with physical environment, probabilistic models are required to reason about them. This is exacerbated by requirements for fault tolerance, when systems need to be analyzed quantitatively for the amount of failure they can tolerate, or for the delays that may appear. As Henzinger and Sifakis [1] point out, introducing probabilities into design theories allows assessing dependability of IT systems in the same manner as commonly practiced in other engineering disciplines.

Generalizing the notion of MTSs to the non-functional analysis of probabilistic systems, the formalism of Interval Markov Chains (IMCs) was introduced in [46]; with notions of satisfaction and refinement generalizing probabilistic bisimulation. Informally, IMCs extend Markov Chains by labeling transitions with *intervals* of allowed probabilities rather than concrete probability values. Implementations of IMCs are Markov Chains (MCs) whose probability distributions match the constraints induced by the intervals. IMCs are known to be an efficient model on which refinement checking can be performed with efficient algorithms from linear algebra. Unfortunately, as we shall now see, the expressive power of IMCs is inadequate to support both conjunction and parallel composition.

Consider the IMCs of Figure 1. S_1 specifies a behaviour of a user of a coffee machine. It prescribes that a typical user orders coffee with milk with probability within $[0, 0.5]$ and orders black coffee with probability in $[0.2, 0.7]$. Customers also buy tea with probability in the interval $[0, 0.5]$. Now the vendor of the machine delivers another specification, S_2, which prescribes that the machine is functioning only if coffee (white or black) is ordered with probability between 0.4 and 0.8. Otherwise, the machine runs out of coffee powder too frequently, or the powder becomes too old. A conjunction of these two models would describe users who have use patterns compatible with this particular machine. In the bottom part of Figure 1 we present the structure of such a conjunction. States $(2, 3)$, $(3, 3)$, and $(4, 2)$ are inconsistent and thus the corresponding probabilities must be zero: $z_3 = z_5 = z_6 = 0$. Now, attempting to express the conjunction $S_1 \wedge S_2$ as an IMC by a simple intersection of bounds gives $0.4 \leq z_1 \leq 0.5$, $0.4 \leq z_2 \leq 0.7$, and $z_4 \leq 0.5$. However, this naive construction is too coarse: whereas $(z_1, z_2, z_3, z_4, z_5, z_6) = (0.5, 0.5, 0, 0, 0, 0)$ satisfies the constraints the resulting overall probability of reaching a state resulting from State 2 of S_2,i.e. $z_1+z_2+z_3 = 1$, violates the upper bound of 0.8 specified in S_2.

Instead the conjunction should require, among others, that $z_1 + z_2 + z_3 \in [0.4, 0.8]$,

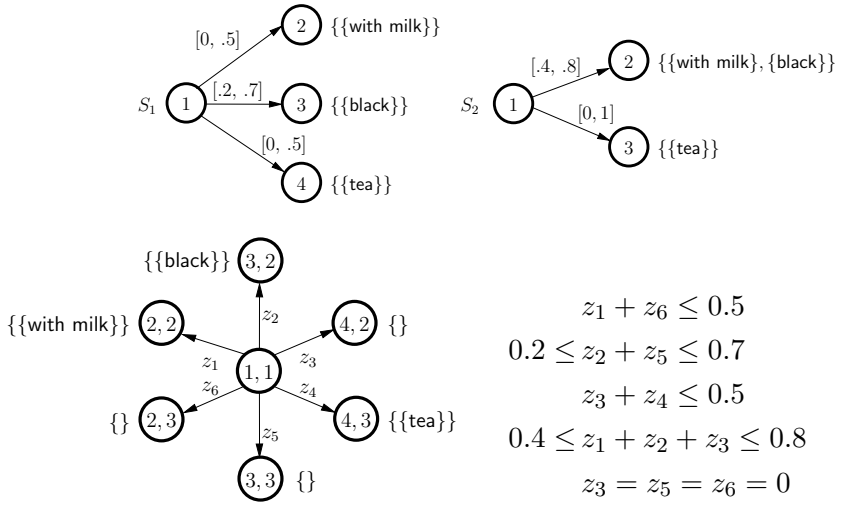


Figure 1: IMCs showing non-closure under conjunction. Top: the two specifications of different aspects of a coffee service. Bottom a conjunction expressed as a Markov Chain with linear constraints over probability values.

which is not an interval constraint. This can be seen be pointing out an extremal point, which is not a solution, while all its coordinates take part in some solution. The bottom right part of Figure 1 lists all the needed constraints over z_i necessary to express conjunction. A similar example can show that IMCs are not closed under parallel composition, either.

One way to approach this problem could be to work with two types of specifications: IMCs for refinement, and with a probabilistic logic such as PCTL [18] on which a logical conjunction is naturally defined. Such a solution is clearly not satisfactory. Indeed, according to [89], there is no procedure to synthesize a MC (an implementation) that satisfies two PCTL formulas in the quantitative case. It is also not possible to structurally compose two logical PCTL formulas.

The solution promoted in this paper is to enrich the model of IMCs. More precisely, we introduce *Constraint Markov Chains* (CMCs) as a foundation for component-based design of probabilistic systems. CMCs are a further extension of IMCs allowing rich constraints on the next-state probabilities from any state. Whereas linear constraints suffice for closure under conjunction, polynomial constraints are necessary for closure under parallel composition. We provide constructs for refinement, consistency checking, conjunction *and* parallel composition of CMC specifications – all indispensable ingredients of a compositional design methodology.

Specification Theories Let us give an overview of specification theories from the point of view of the main operators, and how they are supposed to be used. A more detailed methodological presentation, using the example not of probabilistic, but of timed systems, is available in [90].

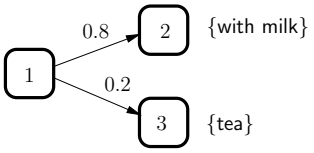


Figure 2: A Markov Chain satisfying specification S_2 of Figure 1

Consistency and Satisfaction. The fundamental notion of a specification theory is *satisfaction*—a relation that binds the specifications to their realizations (*implementations* or models). In our case, specifications are Constraint Markov Chains: mathematical over-approximations of probabilistic behaviours. Implementations are concrete random processes—Markov Chains.

We have shown two examples of IMC specifications in the top of Figure 1: one of expectations of use (S_2) and one of usage in practice (S_1). Observe that these IMCs are also CMCs. In both cases it was easy to see that the specifications are *consistent*, i.e. for each of them one can derive a Markov Chain which satisfies all the interval constraints. For IMC S_2, the example of such an implementation is shown in Figure 2. In development of probabilistic protocols, consistency means that given an abstract specification of a protocol, it is possible to derive a concrete random process satisfying its constraints.

It should be decidable whether a specification admits at least one implementation, and whether a system implements a specification. In our theory, an implementation shall not be viewed as a program in a general purpose programming language but rather as a mathematical object that represents a set of programs sharing common control properties.

Refinement. A *refinement* relation allows to explain whether a given specification S is a proper, consistent elaboration of another more abstract specification T. If this is the case then every implementation of S would also be admitted by T—none of such implementations would violate any requirements of T.

In system development the refinement relation is used to express correctness of a step-wise process, when more coarse-grained descriptions are refined into more detailed ones, until an implementation is obvious. Dually, in verification, refinement is used to establish that an abstract specification is refined by our concrete model. Then the abstract specification, which is usually smaller, can be more efficiently verified for properties preserved by refinement.

In Figure 1 specification S_2 does not refine the specification S_1. This is witnessed by the Markov Chain in Figure 2, which implements S_2, but not S_1.

Parallel composition. A theory should provide a combination operator on specifications, reflecting the standard composition of systems by putting different components together. In the coffee machine example such a situation could arise, if we wanted to analyze the model of a given coffee machine (not shown here), and a given customer, to see whether these two models are compatible. The first step of such an analysis would include combining the two models using parallel composition. Similarly, in protocol

development, parallel composition can be used to combine, for example, specifications of a client and a server.

It is a common modeling scenario to use parallel composition of components, say S_1 and S_2, to build specifications of systems that are supposed to refine general requirements specified using one model, for example T. Then refinement check is performed, $S_1 \parallel S_2 \leq T$, to verify whether indeed a decomposition into components is correct.

Conjunction. In contrast to parallel composition, conjunction is used to combine different specifications for the very same component. Such a need could arise, if the model of the component consists of several separate specifications for different viewpoints, or arising from different stakeholders. The conjunction of two specifications should thus correspond to a specification whose implementations are all implementations of both conjoined specifications.

In our example, we already know that not all implementations of the coffee machine (S_2) are also implementations of S_1—in practical terms there exist machines that have requirements incompatible with requirements of our users. A different question is to ask, whether there exist any machines that are able to work with at least one of our users. This corresponds to asking whether specification $S_1 \wedge S_2$ is consistent. It is not hard to check that this specification, presented as a CMC in Fig. 1, can be satisfied by a Markov Chain, as already discussed.

Incremental Design. A theory should allow incremental design (composing or conjoining specifications in any order) and independent implementability (composable specifications can always be refined separately) [91].

For example it should be possible to create specifications for communicating components S_1 and S_2, and refining them independently, to say P_1 and P_2 respectively, without loosing the overall refinement; so still $P_1 \parallel P_2 \leq S_1 \parallel S_2$.

Detailed Results In the above summary we illustrated the main operators of a specification theory using the IMCs of Figures 1 and 2. However, as argued earlier, such a theory cannot be build for IMCs due to the lack of suitable closure properties. In this paper we develop the theory for Constraint Markov Chains. Below we summarize the most important design decisions. A less experienced reader may choose to skip this rather technical summary during the first reading.

The notions of satisfaction and strong/weak refinements for CMCs conservatively extend similar notions for IMCs [46, 48]. We characterize these relations in terms of implementation set inclusion. In particular, in the main theorem, we prove that for deterministic CMCs weak and strong refinements are complete with respect to implementation set inclusion. In addition, we provide a construction, which for any CMC S returns a deterministic CMC $\rho(S)$ containing the models of S. Refinement relations are not complete for non-deterministic CMCs, but one can show that the weak refinement is more likely to coincide with implementation set inclusion in such a context. We show that refinement between CMCs with polynomial constraints can be decided in essentially single exponential time.

In CMCs, each state is also labeled with a set of subsets of atomic propositions. Those propositions represent properties that should be satisfied by the implementation, the idea being that the satisfaction relation ensures that an implementation matches at least one of the subsets. This allows the specification to make additional abstraction of the behaviors of the implementation. Hence, at the level of specification, our model presents choices on subsets of atomic propositions. However these choices are independent from the probabilistic ones in the sense that any CMC whose states are labeled with a set of subsets of atomic propositions can be turned to an equivalent (in terms of set of implementations) CMC whose states are labeled with a single subset of atomic propositions. There, choices between the subsets of atomic propositions disappear. It is thus not surprising that our notion of parallel composition is following the widely accepted *principle of separation of concerns*. The idea is to separate parallel composition of probability distributions from synchronization on sets of atomic propositions. This separation can be found in probabilistic specification theories that have probabilistic automata as an underlying semantic model [6, 28, 30, 92]. In fact, we show how probabilistic automata can be represented as CMCs, and how the traditional notions of parallel composition on such a model can be derived in our framework. This latter result shows that CMCs capture computational structure of known models and operators, laying down a basis for studying shared properties of many probabilistic automata based languages. We exemplify this by showing how precongruence properties for composition of probabilistic automata and known refinements can be obtained by reductions to CMCs.

We also compare the expressiveness of the operation of parallel composition and the one of conjunction. It turns out that for independent sets of valuations, composition refines conjunction, but the opposite is not true. This result allows to isolate a class of CMCs and CMCs operations that is closed under linear constraints. Finally, we also show that CMCs are generally not closed under disjunction and we discuss the problem of deciding whether a CMC is universal.

Structure of the paper The paper is structured as follows. In Section 4, we introduce the concept of CMCs, satisfaction relation with respect to Markov Chains and the problem of consistency. Refinement is discussed in Section 5 while conjunction is presented in Section 6. Parallel composition is introduced in Section 7, where we also compare the operation to conjunction. Disjunction and universality are discussed in Section 8. In Section 9, we introduce deterministic CMCs and show that, for this class of CMCs, strong and weak refinements coincide with inclusion of implementation sets. Section 10 discusses the class of polynomial CMCs, which is the smallest class of CMCs closed under all the compositional design operations. Section 12 concludes the paper with related and future work.

3 Background Definitions

In this section, we introduce concepts and definitions that will be used through the rest of the paper.

Let A, B be sets of propositions with $A \subseteq B$. The *restriction of* $W \subseteq B$ *to* A is given by $W{\downarrow_A} \equiv W \cap A$. If $T \subseteq 2^B$, then $T{\downarrow_A} \equiv \{W{\downarrow_A} \mid W \in T\}$. For $W \subseteq A$ define the *extension of* W *to* B as $W{\uparrow^B} \equiv \{V \subseteq B \mid V{\downarrow_A} = W\}$, so the set of sets whose restriction to A is W. Lift it to sets of sets as follows: if $T \subseteq 2^A$, then $T{\uparrow^B} \equiv \{W \subseteq B \mid W{\downarrow_A} \in T\}$.

Let $M, \Delta \in [0,1]^{n \times k}$ be two matrices and $x \in [0,1]^k$ be a row vector. We write M_{ij} for the cell in ith row and jth column of M, M_p for the pth row of M, and x_i for the ith element of x. Finally, Δ is a *correspondence matrix* iff $0 \leq \sum_{j=1}^{k} \Delta_{ij} \leq 1$ for all $1 \leq i \leq n$. We define the following operations:

1. If $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{k \times r}$ are two correspondence matrices, we define $\Delta'' = \Delta \otimes \Delta'$ by $\Delta'' \in [0,1]^{k \times (qr)}$ and $\Delta''_{i(j,n)} = \Delta_{ij}\Delta'_{in}$.

2. If $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{r \times s}$ are two correspondence matrices, we define $\Delta'' = \Delta \odot \Delta'$ by $\Delta'' \in [0,1]^{(kr) \times (qs)}$ and $\Delta''_{(i,j)(n,p)} = \Delta_{in}\Delta'_{jp}$.

We have the following lemma.

Lemma 1. *1. Let $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{q \times r}$ be two correspondence matrices. The matrix $\Delta'' = \Delta\Delta'$ is a correspondence matrix;*

2. Let $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{k \times r}$ be two correspondence matrices. The matrix $\Delta'' = \Delta \otimes \Delta'$ is a correspondence matrix;

3. Let $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{r \times s}$ be two correspondence matrices. The matrix $\Delta'' = \Delta \odot \Delta'$ is a correspondence matrix;

Proof. We first prove the upper-bound on the row-sum of each of the matrices.

1. Let $1 \leq i \leq k$ and $1 \leq j \leq r$. We have $\Delta''_{ij} = \sum_{n=1}^{q} \Delta_{in}\Delta'_{nj}$. Thus,

$$\sum_{j=1}^{r} \Delta''_{ij} = \sum_{j=1}^{r}\sum_{n=1}^{q} \Delta_{in}\Delta'_{nj} = \sum_{n=1}^{q}\sum_{j=1}^{r} \Delta_{in}\Delta'_{nj}$$
$$= \sum_{n=1}^{q}(\Delta_{in}\sum_{j=1}^{r} \Delta'_{nj}) \leq \sum_{n=1}^{q} \Delta_{in} \leq 1$$

2. Let $1 \leq i \leq k$ and $(j,n) \in \{1,\ldots,q\} \times \{1,\ldots,r\}$. We have $\Delta''_{i(j,n)} = \Delta_{ij}\Delta'_{in}$. Thus, similarly as above:

$$\sum_{(j,n) \in \{1,\ldots q\} \times \{1,\ldots r\}} \Delta''_{i(j,n)} = \sum_{j=1}^{q}\sum_{n=1}^{r} \Delta_{ij}\Delta'_{in} = \sum_{j=1}^{q}(\Delta_{ij}\sum_{n=1}^{r} \Delta'_{in}) \leq 1$$

3. Let $(i,j) \in \{1,\dots k\} \times \{1,\dots r\}$ and $(n,p) \in \{1,\dots q\} \times \{1,\dots s\}$. We have $\Delta''_{(i,j)(n,p)} = \Delta_{in}\Delta'_{jp}$. Thus,

$$\sum_{(n,p)\in\{1,\dots q\}\times\{1,\dots s\}} \Delta''_{(i,j)(n,p)} = \sum_{n=1}^{q}\sum_{p=1}^{s} \Delta_{in}\Delta'_{jp} = (\sum_{n=1}^{q} \Delta_{in})(\sum_{p=1}^{s} \Delta'_{jp}) \leq 1$$

All three row-sums are non-negative, as all the matrices only contain non-negative numbers. □

4 Constraint Markov Chains

In this section, we explicitly introduce the concept of *Constraint Markov Chains* (CMCs). We first begin with the definition of *Markov Chains* (MCs) that act as models for CMCs.

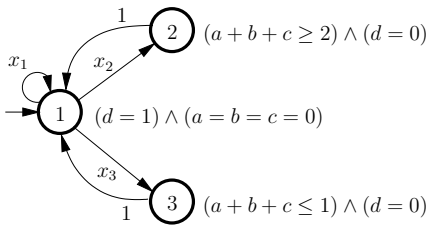
Definition 1 (Markov Chain)**.** *$P = \langle\{1,\dots,n\}, o, M, A, V\rangle$ is a* Markov Chain *if $\{1,\dots,n\}$ is a set of states containing the initial state o, A is a set of atomic propositions, $V : \{1,\dots,n\} \to 2^A$ is a state valuation, and $M \in [0,1]^{n\times n}$ is a probability transition matrix: $\sum_{j=1}^{n} M_{ij} = 1$ for $i = 1,\dots,n$.*

We now introduce *Constraint Markov Chains* (CMCs for short), a finite representation for a possibly infinite set of MCs. Roughly speaking, CMCs generalize MCs in that, instead of specifying a concrete transition matrix, they only constrain probability values in the matrix. Constraints are modeled using a *characteristic function*, which for a given source state and a distribution of probabilities of leaving the state evaluates to 1 iff the distribution is permitted by the specification. Similarly, instead of a concrete valuation function for each state, a *constraint on valuations* is used. Here, a valuation is permitted iff it is contained in the set of admissible valuations of the specification.

Definition 2 (Constraint Markov Chain)**.** *A* Constraint Markov Chain *is a tuple $S = \langle\{1,\dots,k\}, o, \varphi, A, V\rangle$, where $\{1,\dots,k\}$ is a set of states containing the initial state o, A is a set of atomic propositions, $V : \{1,\dots,k\} \to 2^{2^A}$ is a set of admissible state valuations and $\varphi : \{1,\dots,k\} \to [0,1]^k \to \{0,1\}$ is a* constraint function *such that, for all states $1 \leq j \leq k$, if $\varphi(j)(x) = 1$, then the x vector is a probability distribution: $x \in [0,1]^k$ and $\sum_{i=1}^{k} x_i = 1$.*

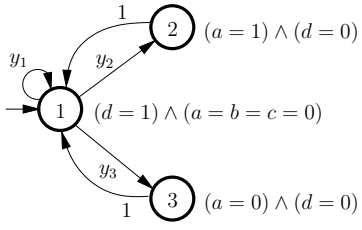
In the rest of the document, we consider that the last constraint, $\sum_{i=1}^{k} x_i = 1$, is implicit and usually dropped in the examples.
An *Interval Markov Chain* (IMC for short) [46] is a CMC whose constraint functions are represented by intervals, so for all $1 \leq i \leq k$ there exist constants α_i, β_i such that, for all states $1 \leq j \leq k$, $\varphi(j)(x) = 1$ iff $\forall 1 \leq i \leq k,\ x_i \in [\alpha_i, \beta_i]$.

Example 1. *Two parties, a customer and a vendor, are discussing a design of a relay for an optical telecommunication network. The relay is designed to amplify an optical signal transmitted over a long distance optical fiber. The relay should have several modes of operation, modeled by four dynamically changing properties and specified by atomic propositions a, b, c, and d:*



$\varphi_1(1)(x) \equiv (x_2 \geq 0.7) \wedge (x_2 + x_3 = 1)$

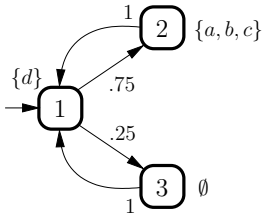
(a) CMC S_1, the customer specification of the optical relay



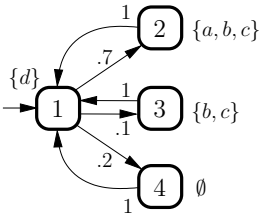
$\varphi_2(1)(y) \equiv (y_3 \geq 0.2) \wedge (y_2 + y_3 = 1)$

(b) CMC S_2, the manufacturer specification of the relay

Figure 3: Two CMCs specifying an optical relay from different perspectives.



(a) Markov Chain P_1 satisfying S_1 and S_2



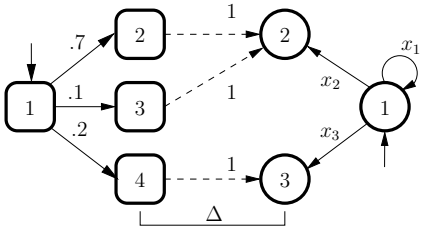
(b) Markov Chain P_2 satisfying S_1 and S_2

Figure 4: Two implementations (MCs) of an optical relay.

Atomic propositions in the optic relay specifications		
a	ber $\leq 10^{-9}$	bit error rate lower than 1 per billion bits transmitted
b	br $>$ 10Gbits/s	The bit rate is higher than 10 Gbits/s.
c	$P <$ 10W	Power consumption is less than 10 W.
d	Standby	The relay is not transmitting.

The customer presents CMC S_1 (Figure 3a) specifying the admissible behaviour of the relay from their point of view. States are labeled with formulas characterizing sets of valuations. For instance, "$(a+b+c\geq 2)\wedge(d=0)$" at state 2 of S_1 represents $V_1(2) = \{\{a,b\},\{b,c\},\{a,c\},\{a,b,c\}\}$, where a, b, c, and d range over Booleans. State 1 specifies a standby mode, where no signal is emitted and only marginal power is consumed. State 2 is the high power mode, offering a high signal/noise ratio, and hence a high bit-rate and low error rate, at the expense of a high power consumption. State 3 is the low power mode, with a low power consumption, low bit-rate and high error rate. The customer prescribes that the probability of the high power mode (state 2) is not less than 0.7*. The vendor replies with CMC S_2 (Figure 3b), which represents possible relays that they can build. Because of thermal limitations, the low power mode has a probability higher than* 0.2*.*

A state u of S is (directly) *reachable* from a state i if there exists a probability distribution $x \in [0,1]^k$ with a nonzero probability x_u, which satisfies $\varphi(i)(x)$.



$$\Delta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 5: Correspondence for initial states of P_2 and S_1

4.1 Satisfaction

We relate CMC specifications to MCs implementing them by extending the definition of satisfaction presented in [46]. The main modification with regard to that definition is the matching of valuation constraints (instead of concrete valuations) and satisfying the full-fledged constraint functions of CMCs (instead of concrete probability distributions). Crucially, just like in [46], we abstract from syntactic structure of transitions—a single transition in the implementation MC can contribute to satisfaction of more than one transition in the specification by distributing its probability mass against several transitions. Similarly, several MC transitions can contribute to satisfaction of a single specification transition.

Definition 3 (Satisfaction Relation)**.** *Let* $P = \langle \{1, \ldots, n\}, o_P, M, A_P, V_P \rangle$ *be a MC and* $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ *be a CMC with* $A_S \subseteq A_P$*. Then* $\mathcal{R} \subseteq \{1, \ldots, n\} \times \{1, \ldots, k\}$ *is a* satisfaction relation *between states of* P *and* S *iff whenever* $p \mathcal{R} u$*, then*

1. $V_P(p)\downarrow_{A_S} \in V_S(u)$*, and*
2. *there exists a correspondence matrix* $\Delta \in [0,1]^{n \times k}$ *such that*
   - *for all* $1 \leq p' \leq n$ *with* $M_{pp'} \neq 0$, $\sum_{j=1}^{k} \Delta_{p'j} = 1$*;*
   - $\varphi(u)(M_p \Delta)$ *holds and*
   - *if* $\Delta_{p'u'} \neq 0$*, then* $p' \mathcal{R} u'$*.*

We write $P \models S$ iff there exists a satisfaction relation relating o_P and o_S, and call P an *implementation* of S. The set of all implementations of S is given by $[\![S]\!] \equiv \{P \mid P \models S\}$. Rows of Δ that correspond to reachable states of P always sum up to 1. This is to guarantee that the entire probability mass of implementation transitions is allocated. For unreachable states, we leave the corresponding rows in Δ unconstrained. P may have a richer set of atomic propositions than S, in order to facilitate abstract modeling: this way an implementation can maintain local information using internal variables. Algorithms to decide satisfaction are particular cases of algorithms to decide *refinement* between CMCs. See the next section.

Example 2. *We illustrate the concept of correspondence matrix between Specification* S_1 *(given in Figure 3a) and Implementation* P_2 *(given in Figure 4b). The CMC* S_1 *has three outgoing transitions from state 1 but, due to constraint function in* 1*, the transition*

labeled with x_1 cannot be taken (the constraint implies $x_1 = 0$). The probability mass going from State 1 to States 2 and 3 in P_2 corresponds to the probability allowed by S_1 from its state 1 to its state 2; The redistribution is done with the help of the matrix Δ given in Figure 5. The ith column in Δ describes how big a fraction of each transition probability (for transitions leaving 1) is associated with probability x_i in S_1. Observe that the constraint function $\varphi_1(1)(0, 0.8, 0.2) = \varphi_1(1)((0, 0.7, 0.1, 0.2)\Delta)$ is satisfied.

CMC semantics follows the Markov Decision Process (MDP) tradition [76, 77]. The MDP semantics is typically opposed to the Uncertain Markov Chain semantics, where the probability distribution from each state is fixed a priori.

States of CMCs are labeled with set of subsets of atomic propositions. A single set of propositions represents properties that should be satisfied by the implementation. A set of sets models a choice of properties, with the idea being that the satisfaction relation ensures that an implementation matches at least one of the subsets.

4.2 Consistency

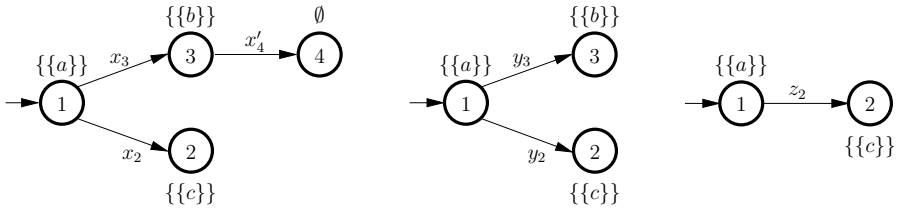
We now define the notion of consistency and propose an algorithm that turns any consistent CMC in a CMC with no inconsistent states.

A CMC S is *consistent* if it admits at least one implementation. We now discuss how to decide consistency. A state u of S is *valuation consistent* iff $V(u) \neq \emptyset$; it is *constraint consistent* iff there exists a probability distribution vector $x \in [0, 1]^k$ such that $\varphi(u)(x) = 1$. It is easy to see that if *each state* of S is both valuation and constraint consistent, then S is also consistent. However, inconsistency of a state, called *local inconsistency*, does not imply inconsistency of the specification, called *global inconsistency*. Indeed, an inconsistent state could be made unreachable by forcing the probabilities to reach it to zero. The operations presented later in this paper may introduce inconsistent states, leaving a question if a resulting CMC is consistent. In order to decide whether S is inconsistent, state inconsistencies are propagated throughout the entire state-space using a *pruning operator* β that removes inconsistent states from S. The result $\beta(S)$ is a new CMC, which may still contain some inconsistent states. We define β formally. Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC.

- If the initial state o is locally inconsistent, then let $\beta(S) = \emptyset$ (meaning that it is not well-defined, returning an empty CMC).

- If S does not contain locally inconsistent states, then $\beta(S) = S$.

- Else proceed in two steps. Let $k' < k$ be the number of locally consistent states. Then define a function $\nu : \{1, \ldots, k\} \to \{\bot, 1, \ldots, k'\}$. All inconsistent states are mapped to $\bot$, i.e. for all $1 \leq i \leq k$ take $\nu(i) = \bot$ iff $[(V(i) = \emptyset) \vee (\forall x \in [0, 1]^k,\ \varphi(i)(x) = 0)]$. All remaining states are mapped injectively into $\{1, \ldots, k'\}$: $\nu(i) \neq \bot \implies \forall j \neq i,\ \nu(j) \neq \nu(i)$. Then let $\beta(S) = \langle \{1, \ldots, k'\}, \nu(o), \varphi', A, V' \}$, where $V'(i) = V(\nu^{-1}(i))$ and for all $1 \leq j \leq k'$ the constraint $\varphi'(j)(y_1, \ldots, y_{k'})$



(a) CMC S before pruning **(b)** CMC $\beta(S)$ **(c)** CMC $\beta^*(S)$

Figure 6: Illustration of the pruning algorithm.

is: $\exists x_1, \ldots, x_k$ such that

$$\big[\nu(q) = \bot \Rightarrow x_q = 0\big] \ \wedge \ \big[\forall 1 \leq l \leq k' : \ y_l = x_{\nu^{-1}(l)}\big] \ \wedge \ \big[\varphi(\nu^{-1}(j))(x_1, \ldots, x_k)\big]$$

The constraint makes the inconsistent states unreachable, and then $\bot$ is dropped as a state. Note that, in practice, the new constraint can be computed in linear time by applying substitutions (of zeros for inconsistent states probabilities) and variable renaming.

The operator is applied iteratively, until a fixpoint is reached. S is consistent if the resulting CMC $\beta^*(S)$ is non-empty. The unique maximum fixpoint is known to exist due to Tarski's theorem, as β is a monotonic decreasing operator on a finite powerset lattice ordered by set inclusion (β operates on sets of states). The following example illustrates the pruning algorithm.

Example 3. *Consider the CMC* $S = \langle \{1,2,3,4\}, 1, \varphi, \{a,b,c\}, V \rangle$ *given in Figure 6a. Define* φ *as follows :* $\varphi(1)(x) \equiv (x_3 \leq 0.3) \wedge (x_2 + x_3 = 1)$, $\varphi(3)(x') \equiv (x'_4 = 1)$. *The constraint of states* 2 *and* 4 *are not relevant for this example.*

State 4 *is obviously not valuation consistent. States* $1, 2$ *and* 3 *are all valuations and constraint consistent. As a consequence, the first step of the pruning algorithm will only mark state* 4 *as inconsistent. Define the following function:*

$$\nu = [1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto \bot] \tag{1}$$

Then define $\beta(S) = \langle \{1,2,3\}, 1, \varphi', \{a,b,c\}, V' \rangle$ *such that, after reduction we have* $\varphi'(1)(y) \equiv (y_3 \leq 0.3) \wedge (y_2 + y_3 = 1)$, *and* $\varphi'(3)(y') \equiv \exists x'_4, \ (x'_4 = 0) \wedge (x'_4 = 1)$. $\beta(S)$ *is given in Figure 6b.*

Obviously, state 3 *of* $\beta(S)$ *is now constraint inconsistent:* $\varphi'(3)(y')$ *is not satisfiable. We thus apply another time the pruning operator* β *in order to remove state* 3. *This time we obtain a consistent CMC* $\beta^*(S)$, *given in Figure 6c.*

Theorem 2. *Let* $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle\}$ *be a CMC and let* $\beta^*(S) = \lim_{n \to \infty} \beta^n(S)$ *be the fixpoint of* β. *For any MC* P, *we have (1)* $P \models S \iff P \models \beta(S)$ *and (2)* $[\![S]\!] = [\![\beta^*(S)]\!]$.

Proof. Let $S = \langle \{1, \dots, k\}, o, \varphi, A, V \rangle$ be a CMC (with at least one inconsistent state) and $P = \langle \{1, \dots, n\}, o_P, M, A_P, V_P \rangle$ be a MC. Let $S' = \beta(S) = \langle \{1, \dots, k'\}, o', \varphi', A, V' \rangle$ be the result of applying the pruning algorithm to S. If $\beta(S)$ is empty, then both S and $\beta(S)$ are inconsistent.

Consider a function ν for removing inconsistent states (one exists because there are inconsistent states), such that $k' < k$ and for all $1 \leq i \leq k$, $\nu(i) = \bot \iff [(V(i) = \emptyset) \vee (\forall x \in [0,1]^k,\ \neg\varphi(i)(x))]$ and $\nu(i) \neq \bot \Rightarrow \forall j \neq i,\ \nu(j) \neq \nu(i)$. We first show that $P \models S \iff P \models \beta(S)$.

$\Rightarrow$ Suppose that $P \models S$. Then there exists a satisfaction relation $\mathcal{R}$ such that $o_P \mathcal{R} o$. Define the relation $\mathcal{R}' \subseteq \{1, \dots, n\} \times \{1, \dots, k'\}$ such that $p \mathcal{R}' v$ iff there exists $u \in \{1, \dots, k\}$ such that $p \mathcal{R} u$ and $\nu(u) = v$. It is clear that $o_P \mathcal{R}' o'$. We prove that $\mathcal{R}'$ is a satisfaction relation. Let p, u, v such that $p \mathcal{R} u$ and $\nu(u) = v$.

- As $\nu(u) \neq \bot$, we have by definition that $V'(v) = V(u)$, thus $V_P(p)\downarrow_A \in V'(v)$.
- Let $\Delta \in [0,1]^{n \times k}$ be the correspondence matrix witnessing $p \mathcal{R} u$. Let $\Delta' \in [0,1]^{n \times k'}$ such that $\Delta'_{qw} = \Delta_{q\nu^{-1}(w)}$. It is clear that Δ' is a correspondence matrix. We first show that

$$\forall u' \in \{1, \dots, k\},\ (\nu(u') = \bot) \Rightarrow (\forall q \in \{1, \dots, n\},\ \Delta_{qu'} = 0). \tag{2}$$

  Let $u' \in \{1, \dots, k\}$ such that $\nu(u') = \bot$, and suppose that there exists $q \in \{1, \dots, n\}$, $\Delta_{qu'} \neq 0$. As Δ is a correspondence matrix, we have $q \mathcal{R} u'$. Thus $V_P(q)\downarrow_A \in V(u')$, which means that $V(u') \neq \emptyset$, and there exists Δ'' such that $\varphi(u')(M_q \Delta'')$. Thus, there exists $x \in [0,1]^k$ such that $\varphi(u')(x)$. As a consequence, we cannot have $\nu(u') = \bot$, which is a contradiction, thus (2).

  We now prove that $\mathcal{R}'$ satisfies the axioms of a satisfaction relation.

  1. Let $p' \in \{1, \dots, n\}$ such that $M_{pp'} \neq 0$. This implies, by definition, that $\sum_{j=1}^{k} \Delta_{p'j} = 1$. We have $\sum_{j=1}^{k'} \Delta'_{p'j} = \sum_{r \in \{1,\dots,k\} \mid \nu(r) \neq \bot} \Delta_{p'r}$. By (2), $\sum_{r \in \{1,\dots,k\} \mid \nu(r) \neq \bot} \Delta_{p'r} = \sum_{r=1}^{k} \Delta_{p'r} = 1$.
  2. Let $y = M_p \Delta' \in [0,1]^{k'}$ and $x = M_p \Delta \in [0,1]^{1 \times k}$. We know that $\varphi(u)(x)$ holds. Moreover, by (2), if $\nu(q) = \bot$, then $x_q = 0$, and for all $l \in \{1, \dots, k'\}$, $y_l = x_{\nu^{-1}(l)}$. Clearly, this implies that $\varphi'(v)(M_p \Delta')$ holds.
  3. Let $p', v' \in \{1, \dots, n\} \times \{1, \dots, k'\}$ such that $\Delta'_{p'v'} \neq 0$. We have $\Delta'_{p'v'} = \Delta_{p'\nu^{-1}(v')} \neq 0$, thus there exists $u' \in \{1, \dots, k\}$ such that $p' \mathcal{R} u'$ and $\nu(u') = v'$. Finally $p' \mathcal{R}' v'$.

  Finally, $\mathcal{R}'$ is a satisfaction relation such that $o_P \mathcal{R}' o'$, thus $P \models \beta(S)$.

$\Leftarrow$ Conversely, the reasoning is the same, except that we now build Δ from Δ' saying that $\Delta_{qv} = 0$ if $\nu(v) = \bot$ and $\Delta_{qv} = \Delta'_{q\nu(v)}$ otherwise.

We have proved that β is implementations-conservative. Thus the fixpoint of β verifies the same property (this can be concluded by mathematical induction, given that the fixpoint is always reached in a finite number of steps). □

The fixpoint of β, and thus the entire consistency check, can be computed using a quadratic number of state consistency checks. The complexity of each check depends on the constraint language that has been chosen.

4.3 Single Valuation Normal Form

It turns out that any CMC whose states are labeled with a set of subsets of atomic propositions can be turned into an equivalent CMC (in terms of sets of implementations) whose states are labeled with sets that contains a single subset of atomic propositions. Hence, working with sets of subsets of valuations is a kind of modeling sugar that can be removed with a transformation to the *single valuation normal form*. We now give details regarding this theory.

Definition 4. *We say that a CMC is in a* Single Valuation Normal Form *if all its admissible valuation sets are singletons ($|V(i) \models 1$ for each $1 \leq i \leq k$).*

More precisely every consistent CMC with at most one admissible valuation in the initial state can be transformed into the normal form preserving its implementation set.

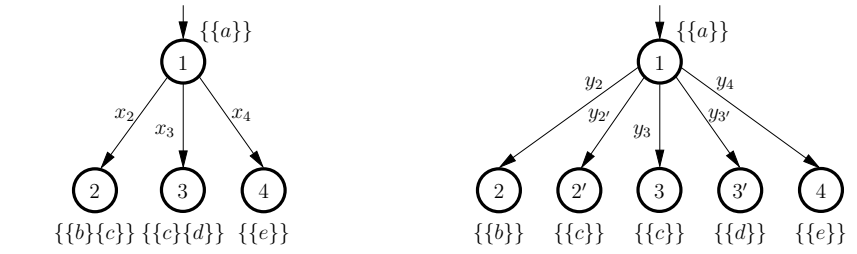
The normalization algorithm, which is presented in Definition 5, basically separates each state u with m possible valuations into m states $u_1, \ldots, u_m$, each with a single admissible valuation. Then the constraint function is adjusted, by substituting sums of probabilities going to the new states in place of the old probabilities targeting u. The transformation is local and syntax based. It can be performed in polynomial time and it only increases the size of the CMC polynomially. We will write $\mathcal{N}(S)$ for a result of normalization of S.

Definition 5 (Normalization algorithm)**.** *Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC. The normalization of S is only defined if o is in single valuation normal form (i.e. $|V(o) \models 1$) and if there exists a function $\mathcal{N} : \{1, \ldots, k\} \to 2^{\{1,\ldots,m\}}$ such that:*

1. *$\{1, \ldots, m\} = \cup_{i \in \{1,\ldots,k\}} \mathcal{N}(i)$;*
2. *For all $1 \leq i \neq j \leq k$, $\mathcal{N}(i) \cap \mathcal{N}(j) = \emptyset$;*
3. *$\forall 1 \leq i \leq k$, $|\mathcal{N}(i)| = |V(i)|$;*

Under these assumptions, the normalization of S is the CMC $\mathcal{N}(S) = \langle \{1, \ldots, m\}, o', \varphi', A, V' \rangle$ such that $\mathcal{N}(o) = o'$ and

1. *$\forall 1 \leq j \leq m$, $|V'(j)| = 1$;*
2. *$\forall 1 \leq i \leq k$, $V(i) = \cup_{u \in \mathcal{N}(i)} V'(u)$;*



(a) CMC S before normalization **(b)** CMC $\mathcal{N}(S)$

Figure 7: Illustration of the normalization algorithm.

3. $\forall 1 \leq i \leq k,\ \forall u, v \in \mathcal{N}(i),\ u \neq v \iff V'(u) \neq V'(v)$;

4. $\forall 1 \leq j \leq m.\ \varphi'(j)(x_1, \dots x_m) = \varphi(\mathcal{N}^{-1}(j))(\sum_{u \in \mathcal{N}(1)} x_u, \dots, \sum_{u \in \mathcal{N}(k)} x_u)$.

By construction, $\mathcal{N}(S)$ is in single valuation normal form. Moreover, if S is consistent, then a function $\mathcal{N}$ satisfying the conditions specified in Definition 5 exists.

The following example illustrates the normalization algorithm.

Example 4. *Consider the CMC* $S = \langle \{1,2,3,4\}, 1, \varphi, \{a,b,c,d,e\}, V \rangle$ *given in Figure 7a. Since states* 2 *and* 3 *have two valuation sets,* S *is not in single valuation normal form. Define the following normalization function:*

$$\mathcal{N} = \left[1 \mapsto \{1\}, 2 \mapsto \{2, 2'\}, 3 \mapsto \{3, 3'\}, 4 \mapsto 4\right] \tag{3}$$

The result of applying the normalization algorithm to S *is the CMC* $\mathcal{N}(S) = \langle \{1, 2, 2', 3, 3', 4\}, 1, \varphi', \{a,b,c,d,e\}, V' \rangle$ *given in Figure 7b. Following the algorithms, states* 2 *and* 3 *of* S *have been each separated into two states with a single valuation. The constraint function of state* 1 *uses* $y_2 + y_{2'}$ *and* $y_3 + y_{3'}$ *instead of* x_2 *and* x_3 *respectively.*

We now show that normalization preserves implementations of a CMC.

Theorem 3. *Let* $S = \langle \{1, \dots k\}, o, \varphi, A, V \rangle$ *be a consistent CMC. If* $|V(o)| = 1$*, then for all MC* P*, we have* $P \models S \iff P \models \mathcal{N}(S)$.

Proof. Let $S = \langle \{1, \dots, k\}, o, \varphi, A, V \rangle$ be a consistent CMC such that $|V(o)| = 1$. Let $S' = \mathcal{N}(S) = \langle \{1, \dots, m\}, o', \varphi', A, V' \rangle$ and $\mathcal{N} : \{1, \dots, k\} \to 2^{\{1,\dots,m\}}$ be the associated function.

($\Rightarrow$) Let $P = \langle \{1, \dots, n\}, o_P, M, A_P, V_P \rangle$ be a MC such that $P \models S$. Let $\mathcal{R}$ be the associated satisfaction relation. Let $\mathcal{R}' \subseteq \{1, \dots, n\} \times \{1, \dots, m\}$ be a new relation such that $p \,\mathcal{R}'\, u \iff V_P(p) \in V'(u)$ and $p \,\mathcal{R}\, \mathcal{N}^{-1}(u)$. We show that $\mathcal{R}'$ is a satisfaction relation. Let p, u such that $p \,\mathcal{R}'\, u$.

1. By definition, we have $V_P(p) \in V'(u)$.
2. We have $p \mathcal{R} \mathcal{N}^{-1}(u)$. Let $\Delta \in [0,1]^{n \times k}$ be the associated correspondence matrix. Define $\Delta' \in [0,1]^{n \times m}$ such that $\Delta'_{q,v} = \Delta_{q,\mathcal{N}^{-1}(v)}$ if $V_p(q) \in V'(v)$ and 0 else. As every coefficient of Δ appears once and only once in the same row of Δ', it is clear that Δ' is a correspondence matrix. Moreover,

   - If q is such that $M_{pq} \neq 0$, then $\sum_{j=1}^{m} \Delta'_{q,j} = \sum_{i=1}^{k} \Delta_{q,i} = 1$;
   - For all $1 \leq i \leq k$, $\sum_{j \in \mathcal{N}(i)}([M_p \Delta']_j) = [M_p \Delta]_i$. As a consequence,

     $$\varphi'(u)(M_p \Delta') = \varphi(\mathcal{N}^{-1}(u))(M_p \Delta)$$

     holds.
   - If q, v are such that $\Delta'_{q,v} \neq 0$, then $\Delta_{q,\mathcal{N}^{-1}(v)} \neq 0$ and $V_P(q) \in V'(v)$, thus $q \mathcal{R}' v$.

Finally, $\mathcal{R}'$ is a satisfaction relation. It is easy to see that $o_p \mathcal{R}' o'$. As a consequence, we have $P \models \mathcal{N}(S)$.

($\Leftarrow$) Let $P = \langle \{1, \ldots, n\}, o_P, M, A_P, V_P \rangle$ be a MC such that $P \models \mathcal{N}(S)$. Let $\mathcal{R}$ be the associated satisfaction relation. Let $\mathcal{R}' \subseteq \{1, \ldots, n\} \times \{1, \ldots, k\}$ such that $p \mathcal{R}' u \iff \exists j \in \mathcal{N}(u)$ s.t. $p \mathcal{R} j$. We will show that $\mathcal{R}'$ is a satisfaction relation. Let p, u such that $p \mathcal{R}' u$.

1. We have $V_P(p) \in V(u) = \cup_{j \in \mathcal{N}(u)} V'(j)$.
2. Let $j \in \mathcal{N}(u)$ such that $p \mathcal{R} j$, and let $\Delta \in [0,1]^{n \times m}$ be the associated correspondence matrix. Define $\Delta' \in [0,1]^{n \times k}$ such that $\Delta'_{q,v} = \sum_{i \in \mathcal{N}(v)} \Delta_{q,i}$. It is clear that for all q, $\sum_{v=1}^{k} \Delta'_{q,v} = \sum_{r=1}^{m} \Delta_{qr}$. Thus Δ' is a correspondence matrix. Moreover,

   - If q is such that $M_{pq} \neq 0$, then $\sum_{i=1}^{k} \Delta'_{q,i} = \sum_{r=1}^{m} \Delta_{q,r} = 1$;
   - For all $1 \leq i \leq k$, $[M_p \Delta']_i = \sum_{r \in \mathcal{N}(i)}([M_p \Delta]_r)$. As a consequence,

     $$\varphi(u)(M_p \Delta) = \varphi'(j)(M_p \Delta')$$

     holds.
   - If q, v are such that $\Delta'_{q,v} \neq 0$, then there exists $r \in \mathcal{N}(v)$ such that $\Delta_{q,r} \neq 0$, thus $q \mathcal{R}' v$.

Finally, $\mathcal{R}'$ is a satisfaction relation. By construction $o_P \mathcal{R}' o$, thus it holds that $P \models S$. □

It is easy to see that normalization preserves determinism.

5 Refinement

Comparing specifications is central to stepwise design methodologies. Systematic comparison enables simplification of specifications (abstraction) and adding details to specifications (elaboration). Usually specifications are compared using a *refinement* relation. Roughly, if S_1 refines S_2, then any model of S_1 is also a model of S_2.

We will now introduce two notions of refinement for CMCs that extend two well known refinements for IMCs [46, 48]. We not only generalize these refinements, but, unlike [46, 48], we also characterize them in terms of implementation set inclusion—also called *thorough refinement*—and computational complexity. We start with the definition of refinements, then we propose algorithms to compute them.

5.1 Refinement Relations

The strong refinement between IMCs, by Jonsson and Larsen [46], extends to CMCs in the following way:

Definition 6 (Strong Refinement)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_2 \subseteq A_1$*. A relation* $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ *is a* strong refinement relation *between states of* S_1 *and* S_2 *iff whenever* $v \,\mathcal{R}\, u$*, then*

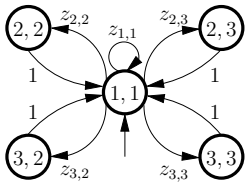
1. $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$*, and*
2. *there exists a correspondence matrix* $\Delta \in [0,1]^{k_1 \times k_2}$ *such that for all probability distribution vectors* $x \in [0,1]^{k_1}$ *if* $\varphi_1(v)(x)$ *holds, then*
   - *for all* $1 \leq i \leq k_1$*,* $x_i \neq 0 \implies \sum_{j=1}^{k_2} \Delta_{ij} = 1$*;*
   - $\varphi_2(u)(x\Delta)$ *holds and*
   - *if* $\Delta_{v'u'} \neq 0$*, then* $v' \,\mathcal{R}\, u'$*.*

We say that S_1 *strongly refines* S_2*, written* $S_1 \preceq_S S_2$*, iff* $o_1 \,\mathcal{R}\, o_2$*.*

Strong refinement imposes a "fixed-in-advance" correspondence matrix Δ regardless of the probability distribution satisfying the constraint function. In contrast, the *weak refinement*, which generalizes the one proposed in [48] for IMCs, allows choosing a different correspondence matrix for each probability distribution satisfying the constraint:

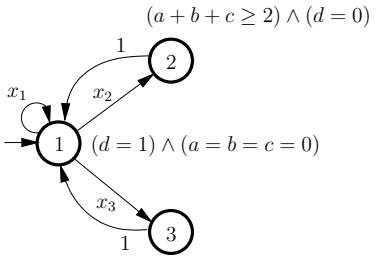
Definition 7 (Weak Refinement)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\},\ o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_2 \subseteq A_1$*. The relation* $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ *is a* weak refinement relation *iff whenever* $v \,\mathcal{R}\, u$*, then:*

1. $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$ *and*
2. *for any distribution* $x \in [0,1]^{k_1}$ *satisfying* $\varphi_1(v)(x)$*, there exists a matrix* $\Delta \in [0,1]^{k_1 \times k_2}$ *such that*



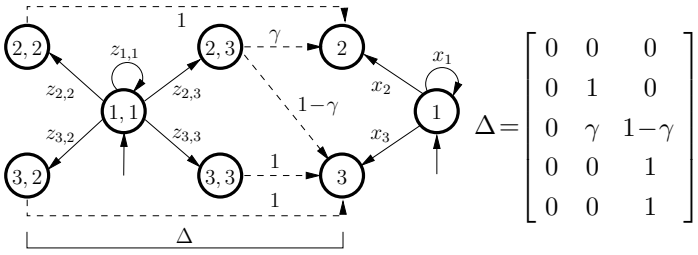
$\varphi_3(1,1)(Z) \equiv (\forall j.\ z_{1,j} = 0)$
$\wedge\ (z_{2,2} + z_{2,3} \geq 0.7)$
$\wedge\ (z_{2,2}+z_{2,3}+z_{3,2}+z_{3,3} = 1)$
$\wedge\ (z_{2,3} + z_{3,3} \geq 0.2)$

(a) $S_3 = S_1 \wedge S_2$. Constraints on propositions, pairwise conjunctions of constraints of S_1 and S_2, are left out to avoid clutter



$\varphi_4(1)(x) \equiv (x_2 \geq 0.7)$
$\wedge (x_3 \geq 0.2) \wedge (x_2 + x_3 = 1)$

(b) CMC S_4 generalizing S_3, so $S_3 \preceq S_4$



(c) Weak refinement for initial states of S_3 and S_4

Figure 8: Examples of refinement and conjunction

- *for all* $1 \leq i \leq k_1$, $x_i \neq 0 \implies \sum_{j=1}^{k_2} \Delta_{ij} = 1$;
- $\varphi_2(u)(x\Delta)$ *holds and*
- *if* $\Delta_{v'u'} \neq 0$, *then* $v' \mathcal{R} u'$.

CMC S_1 *(weakly) refines* S_2, *written* $S_1 \preceq S_2$, *iff* $o_1 \mathcal{R} o_2$.

Example 5. *Figure 8c illustrates a family of correspondence matrices parametrized by* γ, *witnessing the weak refinement between initial states of* S_3 *and* S_4 *(defined in Figures 8a–8b). The actual matrix used in proving the weak refinement depends on the probability distribution vector* z *that satisfies the constraint function* φ_3 *of state* $(1,1)$. *Take* $\gamma = \frac{0.7 - z_{2,2}}{z_{2,3}}$ *if* $z_{2,2} \leq 0.7$ *and* $\gamma = \frac{0.8 - z_{2,2}}{z_{2,3}}$ *otherwise. It is easy to see that* $\varphi_3((1,1))(z)$ *implies* $\varphi_4(1)(z\Delta)$.

Clearly, the existence a strong refinement relation implies the existence of a weak refinement relation, as every strong refinement is a also a weak refinement. Furthermore, both weak and strong refinements imply implementation set inclusion, as shown by the following theorem:

Theorem 4. *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be two CMCs. Assume* $S_1 \preceq S_2$*, we prove that* $[\![S_1]\!] \subseteq [\![S_2]\!]$.

Proof. Since $S_1 \preceq S_2$, there exists a weak refinement relation $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ such that $o_1 \mathcal{R} o_2$. Consider $P = \langle \{1, \ldots n\}, o_P, M, A_P, V_P \rangle$ such that $P \models S_1$. By definition there exists a satisfaction relation $\mathcal{R}' \subseteq \{1, \ldots, n\} \times \{1, \ldots, k_1\}$ such that $o_P \mathcal{R}' o_1$.

Consider the relation $\mathcal{R}'' \subseteq \{1, \ldots, n\} \times \{1, \ldots, k_2\}$, such that $p \mathcal{R}'' u$ iff. there exists $v \in \{1, \ldots, k_1\}$ such that $p \mathcal{R}' v$ and $v \mathcal{R} u$. We prove that $\mathcal{R}''$ is a satisfaction relation. First, it is clear that $A_2 \subseteq A_1 \subseteq A_P$. Now, consider p, u such that $p \mathcal{R}'' u$, so there exists v such that $p \mathcal{R}' v$ and $v \mathcal{R} u$. Since $V_P(p)\downarrow_{A_1} \in V_1(v)$ and $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$, we have that $V_P(p)\downarrow_{A_2} \in V_2(u)$.

We now build a correspondence matrix Δ''. Consider the pth row of M, $M_p \in [0,1]^n$. Let $\Delta' \in [0,1]^{n \times k_1}$ be a correspondence matrix witnessing $p \mathcal{R}' v$. Let $y = M_p \Delta' \in [0,1]^{k_1}$. By Definition 3 we have $\varphi_1(v)(y)$. Let $\Delta \in [0,1]^{k_1 \times k_2}$ be the correspondence matrix witnessing $v \mathcal{R} u$ and define $\Delta'' = \Delta' \Delta \in [0,1]^{n \times k_2}$. By Lemma 1, Δ'' is also a correspondence matrix. We prove that Δ'' satisfies the axioms of Definition 3.

1. Let $1 \leq p' \leq n$ such that $M_{pp'} \neq 0$. As a consequence, $\sum_{q=1}^{k_1} \Delta'_{p'q} = 1$. We want to prove that $\sum_{j=1}^{k_2} \Delta''_{p'j} = 1$.

$$\sum_{j=1}^{k_2} \Delta''_{p'j} = \sum_{j=1}^{k_2} \left(\sum_{q=1}^{k_1} \Delta'_{p'q} \Delta_{qj} \right) = \sum_{q=1}^{k_1} \left(\Delta'_{p'q} \sum_{j=1}^{k_2} \Delta_{qj} \right)$$

   Let q such that $\Delta'_{p'q} \neq 0$. It is then clear that $y_q \geq M_{pp'} \Delta'_{p'q} > 0$. As Δ is a witness of $v \mathcal{R} u$, we have, by the definition of weak refinement, $\sum_{j=1}^{k_2} \Delta_{qj} = 1$. Finally, this implies that $\sum_{j=1}^{k_2} \Delta''_{p'j} = 1$.

2. By Definition 7, since $\varphi_1(v)(M_p \Delta)$ holds, then $\varphi_2(u)(M_p \Delta'')$ holds.

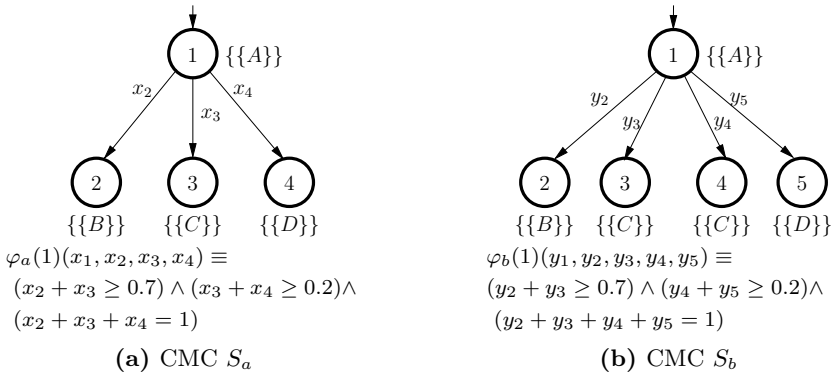
3. Let p', u' such that $\Delta''_{p'u'} \neq 0$. By construction, it is clear that there exists v' such that $\Delta'_{p'v'} \neq 0$ and $\Delta_{v'u'} \neq 0$. By definition of Δ' and Δ, this implies that $p' \mathcal{R}' v'$ and $v' \mathcal{R} u'$, thus $p' \mathcal{R}'' u'$.

From 1-3, we can conclude that $\mathcal{R}''$ is a satisfaction relation. Since $o_P \mathcal{R}'' o_2$, we have $P \in [\![S_2]\!]$ and $[\![S_1]\!] \subseteq [\![S_2]\!]$. □

In Section 9, we shall see that the converse holds for a particular class of CMCs. However, this is not the case in general: strong refinement is strictly stronger than weak refinement, which is strictly stronger than implementation set inclusion. Formally, we have the following proposition.

Proposition 5. *There exist CMCs* S_a, S_b, S_c *and* S_d *such that*

- S_a *weakly refines* S_b*, and* S_a *does not strongly refine* S_b*;*



(a) CMC S_a **(b)** CMC S_b

Figure 9: Distinguishing weak and strong refinement. S_a weakly, but not strongly, refines S_b. Here, and elsewhere, self-targeting loop transitions with probability 1 have been elided (in states with no out-going arrows).

$$\Delta_x = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \gamma & (1-\gamma) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \Delta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & a & (1-a) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

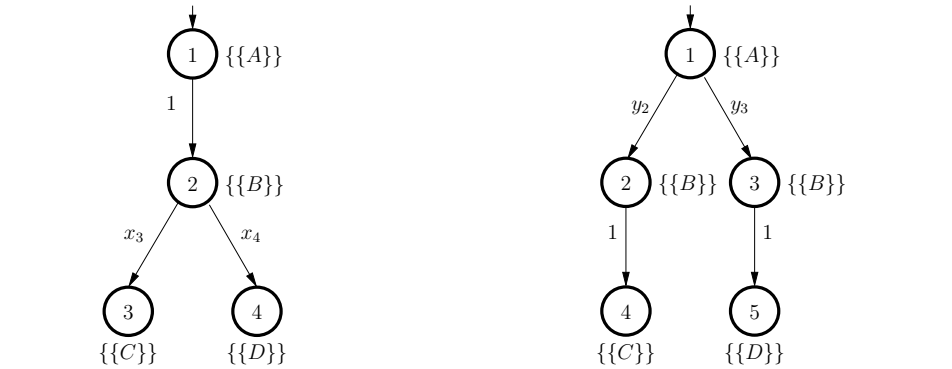
Figure 10: Correspondence matrices for $S_a \preceq S_b$

- $[\![S_c]\!] \subseteq [\![S_d]\!]$, *and* S_c *does not weakly refine* S_d.

Proof. • Consider the CMCs S_a and S_b given in Figures 9a and 9b, respectively. By x_a (resp. y_b) we denote state x in S_a (resp. y in S_b). We first show that there exists a weak refinement relation $\mathcal{R}$ such that $S_a \preceq S_b$, with $1_a \mathcal{R} 1_b$. We then show that there exists no strong refinement relation between S_a and S_b.

1. Let $\mathcal{R} = \{(1_a, 1_b), (2_a, 2_b), (3_a, 3_b), (3_a, 4_b), (4_a, 5_b)\}$. We show that $\mathcal{R}$ is a weak refinement relation. We first focus on building the correspondence matrix for the pair $(1_a, 1_b)$. Let x be a be a distribution satisfying the constraint in 1_a. Let $\gamma = \frac{0.7-x_2}{x_3}$ if $x_2 \leq 0.7$ and $\frac{0.8-x_2}{x_3}$ otherwise. As x satisfies $\varphi_a(1_a)$, we have $0 \leq \gamma \leq 1$. Consider the correspondence matrix Δ_x given in Figure 10

   It is easy to see that for all valuations x satisfying $\varphi_a(1_a)$, $\varphi_b(1_b)(x\Delta_x)$ also holds. The correspondence matrices for the other pairs in $\mathcal{R}$ are trivial. Thus $\mathcal{R}$ is a weak refinement relation between S_a and S_b.
2. Suppose that there exists a strong refinement relation $\mathcal{R}'$ such that $1_a \mathcal{R}' 1_b$. Let Δ be the correspondence matrix witnessing $1_a \mathcal{R}' 1_b$. Since $2_a, 3_a$ and



(a) CMC S_c **(b)** CMC S_d

Figure 11: Weak refinement vs model inclusion: $[\![S_c]\!] \subseteq [\![S_d]\!]$ but $S_c \not\preceq S_d$.

4_a can all be reached from 1_a with an admissible transition, the sum of the elements in the corresponding rows in Δ must be one. From the valuations of the states, we obtain that Δ is of the type given in Figure 10, with $0 \leq a \leq 1$.

Moreover, if $\mathcal{R}'$ is a strong refinement relation, then we have that for all valuation x satisfying $\varphi_a(1_a)$, $\varphi_b(1_b)(x\Delta)$ also holds.

Let $x^1 = (0, 0.6, 0.1, 0.3)$ and $x^2 = (0, 0.8, 0.1, 0.1)$. Both x^1 and x^2 satisfy $\varphi_a(1_a)$. If there exists a strong refinement, this implies that $\varphi_b(1_b)(x^1\Delta)$ and $\varphi_b(1_b)(x^2\Delta)$ also hold. However, $\varphi_b(1_b)(x^1\Delta) = 1$ implies that $a \geq 1$ and $\varphi_b(1_b)(x^2\Delta)$ implies that $a \leq 0$.

It is thus impossible to find a unique correspondence matrix working for all the "valid" valuations of the outgoing transitions of 1_a. As a consequence, there cannot exist a strong refinement relation $\mathcal{R}'$ such that $1_a \,\mathcal{R}'\, 1_b$.

- Consider the CMCs S_c and S_d given in Figures 11a and 11b. It is easy to see that S_c and S_d share the same set of implementations. However, due to the constraints, state 2 of S_c cannot refine any state of S_d. As a consequence, S_c cannot refine S_d.

□

So our refinement relations for CMCs can be ordered from finest to coarsest: the strong refinement, the weak refinement, and the implementation set inclusion. As the implementation set inclusion is the *ultimate* refinement, checking finer refinements is used as a pragmatic syntax-driven, but sound, way of deciding it.

As we shall see in the next section, the algorithms for checking weak and strong refinements are polynomial in the number of states, but the treatment of each state depends on the complexity of the constraints. For the case of implementation set inclusion, the algorithm is exponential in the number of states. Checking implementation

set inclusion seems thus harder than checking weak or strong refinement. In Section 9, we will propose a class of CMCs for which strong and weak refinements coincide with implementation set inclusion.

5.2 Algorithms for Computing Refinements

We now discuss algorithms for checking implementation set inclusion and refinements.

We start with algorithms for checking weak and strong refinements between two CMCs $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ with $k_1, k_2 \leq n$. Checking whether a relation $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ is a strong (resp. weak) refinement relation reduces to checking, for all $(i, j) \in \mathcal{R}$, the validity of the following *refinement formulas*: $\exists \Delta, \forall x, \varphi_1(i)(x) \Rightarrow \varphi_2(j)(x\Delta) \wedge \bigwedge_{i'} (\sum_{j'} \Delta_{i'j'} = 1) \wedge \bigwedge_{i',j'} (i' \mathcal{R} j' \vee \Delta_{i'j'} = 0)$ for the strong refinement, and $\forall x, \varphi_1(i)(x) \Rightarrow \exists \Delta, \varphi_2(j)(x\Delta) \wedge \bigwedge_{i'} (\sum_{j'} \Delta_{i'j'} = 1) \wedge \bigwedge_{i',j'} (i' \mathcal{R} j' \vee \Delta_{i'j'} = 0)$ for the weak refinement. Strong and weak refinements can be decided by iterated strengthening of $\mathcal{R}$ with refinement formulas, starting from $\mathcal{R}_0 = \{(i, j) | V_1(i) \downarrow_{A_2} \subseteq V_2(j)\}$, until either $(o_1, o_2) \notin \mathcal{R}$, in which case S_1 does not strongly (resp. weakly) refine S_2, or $\mathcal{R}$ is found to be a strong (resp. weak) refinement.

The exact complexity of the algorithm depends on the type of constraints that are used in the specifications. As an example, consider that all the constraints in S_1 and S_2 are polynomials of degree d with less than k bound variables – we shall see that polynomial constraints is the smallest class under which CMCs are closed. There, deciding refinement formulas can be done by quantifier elimination. When the number of quantifier alternations is constant, the *cylindrical algebraic decomposition algorithm* [93, 94], implemented in Maple [95], performs this quantifier elimination in time double exponential in the number of variables. Consequently, refinement can be checked in $O(n^2 2^{2^{n^2}})$ time.

However, considering constraints φ which contain only existential quantifiers, quantifier alternation is either one or two for strong refinement and exactly one for weak refinement. There are quantifier elimination algorithms that have a worst case complexity of a single exponential only in the number of variables, although they are double exponential in the number of quantifier alternations [96]. Thanks to these algorithms, deciding whether $\mathcal{R}$ is a strong (resp. weak) refinement relation can be done in time single exponential in the number of states n and k, the number of bound variables appearing in the constraints: $O(n^2 s^{P(n,k)} d^{P(n,k)})$, where P is a polynomial.

We now turn to the case of implementation set inclusion. In [46], Larsen and Jonsson proposed an algorithm for solving this problem for the case of IMCs. This algorithm directly extends to CMCs. The main difference with the algorithms for solving weak and strong refinements is that the algorithm for implementation set inclusion is exponential in the number of states.

Finally, let us mention that lower-bounds for the strong and weak refinement checking remain open problems. On the other hand, in [61], we have shown that implementation set inclusion is EXPTIME-hard for IMCs, hence providing a lower bound also

for CMCs.

6 Conjunction

Conjunction combines requirements of several specifications.

Definition 8 (Conjunction)**.** *Let* $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ *and* $S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ *be two CMCs. The conjunction of* S_1 *and* S_2*, written* $S_1 \wedge S_2$*, is the CMC* $S = \langle \{1,\dots,k_1\} \times \{1,\dots,k_2\}, (o_1,o_2), \varphi, A, V\rangle$ *with* $A = A_1 \cup A_2$*,* $V((u,v)) = V_1(u)\uparrow^A \cap V_2(v)\uparrow^A$*, and*

$$\varphi((u,v))(x_{1,1},x_{1,2},\dots,x_{2,1},\dots,x_{k_1,k_2}) \equiv \begin{array}{l}\varphi_1(u)(\sum_{j=1}^{k_2} x_{1,j},\dots,\sum_{j=1}^{k_2} x_{k_1,j})\wedge\\ \varphi_2(v)(\sum_{i=1}^{k_1} x_{i,1},\dots,\sum_{i=1}^{k_1} x_{i,k_2}).\end{array}$$

Conjunction may introduce inconsistent states. Indeed, the intersection between the sets of valuations of two states may be empty (see state $(2,3)$ in Figure 1). Conjunction should thus normally be followed by applying the pruning operator β^*. As already stated in the introduction, the result of conjoining two IMCs is not an IMC in general, but a CMC whose constraint functions are systems of linear inequalities. Figure 8a depicts a CMC S_3 expressing the conjunction of IMCs S_1 and S_2 (see Figures 3a–3b). The constraint $z_{2,3} + z_{3,3} \geq 0.2$ in state $(1,1)$ cannot be expressed as an interval.

As expected, conjunction of two specifications coincides with their greatest lower bound with respect to the weak refinement (also called *shared refinement*).

Theorem 6. *Let* S_1*,* S_2 *and* S_3 *be three CMCs. We have (a)* $((S_1 \wedge S_2) \preceq S_1)$ *and* $((S_1 \wedge S_2) \preceq S_2)$ *and (b) if* $(S_3 \preceq S_1)$ *and* $(S_3 \preceq S_2)$*, then* $S_3 \preceq (S_1 \wedge S_2)$*.*

Proof. We separately prove the two items of the theorem. Let $\{1,\dots,k_1\}$, $\{1,\dots,k_2\}$, and $\{1,\dots,k_3\}$ be the sets of states of S_1, S_2, and S_3, respectively.

(a) Let $S_1 \wedge S_2 = S = \langle \{1,\dots,k_1\} \times \{1,\dots,k_2\}, o, \varphi, A, V\rangle$. Let $\mathcal{R} \subseteq (\{1,\dots,k_1\} \times \{1,\dots,k_2\}) \times \{1,\dots,k_1\}$ such that $(u,v)\,\mathcal{R}\,w \iff u = w$. We will prove that $\mathcal{R}$ is a strong refinement relation. Let $u \in \{1,\dots,k_1\}$ and $v \in \{1,\dots,k_2\}$. We have $(u,v)\,\mathcal{R}\,u$.

1. By definition of S obtain $V((u,v))\downarrow_{A_1} = (V_1(u)\uparrow^A \cap V_2(v)\uparrow^A)\downarrow_{A_1} \subseteq V_1(u)$.
2. Let $\Delta \in [0,1]^{(k_1k_2)\times k_1}$ such that $\Delta_{(i,j),i} = 1$ and $\Delta_{(i,j),k} = 0$ if $k \neq i$. We now prove that it satisfies the axioms of a satisfaction relation for $(u,v)\,\mathcal{R}\,u$ stated in Definition 3:
   - Then we have $\forall (i,j).\ \sum_{k=1}^{k_1} \Delta_{(i,j),k} = 1$.
   - If $x \in [0,1]^{(k_1k_2)}$ is such that $\varphi((u,v))(x)$, it implies by definition that $\varphi_1(u)(\sum_{j=1}^{k_2} x_{1,j},\dots,\sum_{j=1}^{k_2} x_{k_1,j}) = \varphi_1(u)(x\Delta)$ holds.
   - If $\Delta_{(u',v'),w'} \neq 0$, we have by definition $u' = w'$ and $(u',v')\,\mathcal{R}\,u'$.

We conclude that $\mathcal{R}$ is a strong, and thus also a weak, refinement relation. Since $(o_1, o_2)\,\mathcal{R}\,o_1$, we have $S_1 \wedge S_2 \preceq S_1$. By symmetry, we also have $S_1 \wedge S_2 \preceq S_2$.

(b) Assume $S_3 \preceq S_1$ and $S_3 \preceq S_2$. By definition, there exist refinement relations $\mathcal{R}_1 \subseteq \{1,\ldots,k_3\} \times \{1,\ldots,k_1\}$ and $\mathcal{R}_2 \subseteq \{1,\ldots,k_3\} \times \{1,\ldots,k_2\}$ such that $o_3\,\mathcal{R}_1\,o_1$ and $o_3\,\mathcal{R}_2\,o_2$. Let $S_1 \wedge S_2 = S = \langle \{1,\ldots,k_1\} \times \{1,\ldots,k_2\}, o, \varphi, A, V\rangle$.

Let $\mathcal{R} \subseteq \{1,\ldots,k_3\} \times (\{1,\ldots,k_1\} \times \{1,\ldots,k_2\})$ such that $u\,\mathcal{R}(v,w) \iff u\,\mathcal{R}_1\,v$ and $u\,\mathcal{R}_2\,w$. We now prove that $\mathcal{R}$ is a weak refinement relation.

Consider u, v, w such that $u\,\mathcal{R}(v,w)$.

1. By definition, we have $V_3(u)\downarrow_{A_1} \subseteq V_1(v)$ and $V_3(u)\downarrow_{A_2} \subseteq V_2(w)$. As a consequence, $V_3(u)\downarrow_A \subseteq V((v,w))$.

2. Let $x \in [0,1]^{k_3}$ such that $\varphi_3(u)(x)$. Consider the correspondence matrices $\Delta \in [0,1]^{k_3 \times k_1}$ and $\Delta' \in [0,1]^{k_3 \times k_2}$ given by $u\,\mathcal{R}_1\,v$ and $u\,\mathcal{R}_2\,w$ for the transition vector x. Let $\Delta'' \in [0,1]^{k_3 \times (k_1 k_2)}$ be a new matrix such that $\Delta'' = \Delta \otimes \Delta'$. By Lemma 1, Δ'' is a correspondence matrix. We now prove that it satisfies the axioms of a refinement relation for $u\,\mathcal{R}(v,w)$:

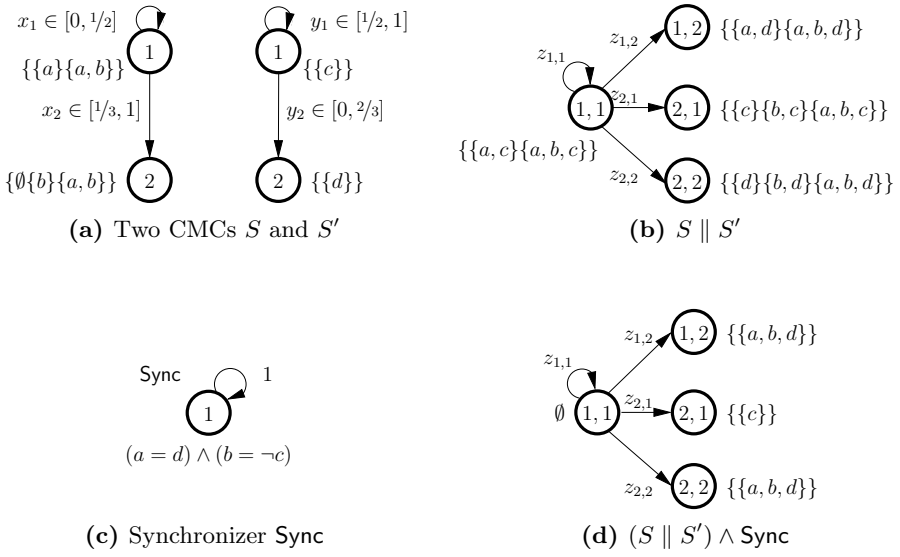
   - Let $1 \leq i \leq k_3$ such that $x_i \neq 0$. By definition of Δ and Δ', we have $\sum_{j=1}^{k_1} \Delta_{ij} = 1$ and $\sum_{q=1}^{k_2} \Delta'_{iq} = 1$, so

$$\begin{aligned}\sum_{(j,q)\in\{1,\ldots,k_1\}\times\{1,\ldots,k_2\}} \Delta''_{i(j,q)} &= (\sum_{j=1}^{k_1} \Delta_{ij})(\sum_{q=1}^{k_2} \Delta'_{iq}) \\ &= 1.\end{aligned}$$

   - By definitions of Δ and Δ', both $\varphi_1(v)(x\Delta)$ and $\varphi_2(w)(x\Delta')$ hold. Let $x' = x\Delta''$. It is clear that $x\Delta = (\sum_{j=1}^{k_2} x'_{(1,j)}, \ldots, \sum_{j=1}^{k_2} x'_{(k_1,j)})$ and $x\Delta' = (\sum_{i=1}^{k_1} x'_{(i,1)}, \ldots, \sum_{i=1}^{k_1} x'_{(i,k_2)})$. As a consequence, $\varphi((v,w))(x\Delta'')$ holds.
   - Let u', v', w' such that $\Delta''_{u'(v',w')} \neq 0$. By construction, this implies $\Delta_{u'v'} \neq 0$ and $\Delta'_{u'w'} \neq 0$. As a consequence, $u'\,\mathcal{R}_1\,v'$ and $u'\,\mathcal{R}_2\,w'$, thus $u'\,\mathcal{R}(v',w')$.

We conclude that $\mathcal{R}$ is a weak refinement relation. Since $o_3\,\mathcal{R}(o_1,o_2)$, we have $S_3 \preceq (S_1 \wedge S_2)$. □

The first consequence of the above theorem is that conjunction with another specification is a monotonic operator with respect to weak refinement. Furthermore, as it follows from the later results of Section 9, the set of implementations of a conjunction of two *deterministic* specifications S_1 and S_2 coincides with the intersection of implementation sets of S_1 and S_2 (the greatest lower bound in the lattice of implementation sets).



(a) Two CMCs S and S' **(b)** $S \parallel S'$

(c) Synchronizer Sync **(d)** $(S \parallel S') \wedge$ Sync

Figure 12: Parallel composition and synchronization of CMCs.

7 Separation of Concerns in Parallel Composition of Specifications

Let us now turn to parallel composition. We first remark, that in concurrency theory it is customary to combine parallel composition with synchronization—for example in many process algebras the same rules are used to explain how parallel processes evolve, and how they communicate. In this work we take a different approach, separating these two concerns— parallel composition from synchronization. The choices regarding sets of valuations and the stochastic choices are independent from each other.

First, components are composed into a kind of product—effectively just a vector of stochastically independent entities. Second, the product is synchronized on valuations by constraining its behaviour, to model handshake communication. For example, assume that one component should be in a state where b holds, whenever $\overline{b}$ holds in another one. An independent product of these two components will likely contain states where only one of the propositions is present, but not the other. Then a synchronization operator is applied that will eliminate these states from the product, ensuring that the two propositions b and $\overline{b}$ always co-occur.

This design has two significant advantages. First, it allows modeling very diverse synchronization mechanisms. For CMCs synchronization goes far beyond matching label names, like in the above example. The synchronization is specified as a part of the model, where it can express complex propositional constraints between propositions (for example a safety property). It can be stateful and probabilistic.

Second, as we will see, we will obtain synchronization by simply using conjunction. This very elegantly exploits the prior results on conjunction, as the synchronization

operator turns out to be realizable using conjunction.

Remark 1. *The principle of separation of concerns is intensively used in the definition of parallel composition for many systems that mix stochastic and non-deterministic choices, among them many theories for probabilistic process algebra [6, 30]. Similar principles also apply for continuous time stochastic models, in a slightly different setting based on CTMCs [92]. In Section 11, in order to argue that such a design is reasonable, we will show that our parallel composition covers the one of probabilistic automata [6]—which is a widely accepted and appreciated operator.*

We start by showing how systems and specifications are composed in a nonsynchronizing manner, then we introduce synchronization. The non-synchronizing *independent* parallel composition is largely just a product of two MCs (or CMCs):

Definition 9 (Parallel Composition of MCs)**.** *Let* $P_1 = \langle \{1, \dots, n_1\}, o_1, M', A_1, V_1 \rangle$ *and* $P_2 = \langle \{1, \dots, n_2\}, o_2, M'', A_2, V_2 \rangle$ *be two MCs with* $A_1 \cap A_2 = \emptyset$*. The parallel composition of* P_1 *and* P_2 *is the MC* $P_1 \parallel P_2 = \langle \{1, \dots, n_1\} \times \{1, \dots, n_2\}, (o_1, o_2), M, A_1 \cup A_2, V \rangle$*, where:* $M \in [0,1]^{(n_1 n_2) \times (n_1 n_2)}$ *is such that* $M = M' \odot M''$ *and* $V((p,q)) = V_1(p) \cup V_2(q)$*.*

For CMCs we have the following definition.

Definition 10 (Parallel Composition of CMCs)**.** *Let* $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_1 \cap A_2 = \emptyset$*. The parallel composition of* S_1 *and* S_2 *is the CMC* $S_1 \parallel S_2 = \langle \{1, \dots, k_1\} \times \{1, \dots, k_2\}, (o_1, o_2), \varphi, A_1 \cup A_2, V \rangle$*, where*

$$\varphi((u,v))(z_{1,1}, z_{1,2}, \dots, z_{2,1}, \dots, z_{k_1,k_2}) \equiv$$
$$\exists x_1, \dots, x_{k_1}, y_1, \dots, y_{k_2} \in [0,1]. \forall (i,j) \in \{1, \dots, k_1\} \times \{1, \dots, k_2\}.$$
$$z_{i,j} = x_i y_j \text{ and } \varphi_1(u)(x_1, \dots, x_{k_1}) = \varphi_2(v)(y_1, \dots, y_{k_2}) = 1$$

Finally, $V((u,v)) = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$*.*

In the introduction we have demonstrated that IMCs are not closed under conjunction. Here, it is worth mentioning that IMCs are not closed under parallel composition, even under the independent one. Consider IMCs S and S' given in Figure 12a and their parallel composition $S \parallel S'$ given in Figure 12b. Assume first that $S \parallel S'$ is an IMC. As a variable $z_{i,j}$ is the product of two variables x_i and y_j, if $S \parallel S'$ is an IMC, then one can show that the interval for $z_{i,j}$ is obtained by computing the products of the bounds of the intervals over which x_i and y_j range. Hence, we can show that $z_{1,1} \in [0, 1/2], z_{1,2} \in [0, 1/3], z_{2,1} \in [1/6, 1], z_{2,2} \in [0, 2/3]$. Let $[a,b]$ be the interval for the constraint $z_{i,j}$, it is easy to see that there exist implementations I_1 of S_1 and I_2 of S_2 such that $I_1 \parallel I_2$ satisfies the constraint $z_{i,j} = a$ (resp. $z_{i,j} = b$). However, while each bound of each interval can be satisfied independently, some points in the polytope defined by the intervals and the constraint $\sum z_{i,j} = 1$ cannot be reached. As an example, consider $z_{1,1} = 0, z_{1,2} = 1/3, z_{2,1} = 1/3, z_{2,2} = 1/3$. It is clearly inside the polytope, but one cannot find an implementation I of $S \parallel S'$ satisfying the constraints given by the parallel composition. Indeed, having $z_{1,1} = 0$ implies that $x_1 = 0$ and thus that $z_{1,2} = 0$.

Theorem 7. *If S_1', S_2', S_1, S_2 are CMCs, then $S_1' \preceq S_1$ and $S_2' \preceq S_2$ implies $S_1' \parallel S_2' \preceq S_1 \parallel S_2$, so the weak refinement is a precongruence with respect to parallel composition. Consequently, for any MCs P_1 and P_2 we have that $P_1 \models S_1 \wedge P_2 \models S_2$ implies $P_1 \parallel P_2 \models S_1 \parallel S_2$.*

Proof. Let

$$\begin{aligned}
S_1 &= \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle,\\
S_2 &= \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle,\\
S_1' &= \langle \{1,\dots,k_1'\}, o_1', \varphi_1', A_1', V_1'\rangle,\\
S_2' &= \langle \{1,\dots,k_2'\}, o_2', \varphi_2', A_2', V_2'\rangle,\\
S &= \langle \{1,\dots,k_1\}\times\{1,\dots,k_2\}, (o_1,o_2), \varphi, A, V\rangle = S_1 \parallel S_2,\\
S' &= \langle \{1,\dots,k_1'\}\times\{1,\dots,k_2'\}, (o_1',o_2'), \varphi', A', V'\rangle = S_1' \parallel S_2',
\end{aligned}$$

be CMCs with $A = A_1 \cup A_2$ and $A' = A_1' \cup A_2'$. Assume that $S_1' \preceq S_1$ and $S_2' \preceq S_2$.

By definition, there exist two weak refinement relations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $o_1' \,\mathcal{R}_1\, o_1$ and $o_2' \,\mathcal{R}_2\, o_2$. Define $\mathcal{R}$ such that $(u',v')\,\mathcal{R}(u,v) \iff u' \,\mathcal{R}_1\, u$ and $v' \,\mathcal{R}_2\, v$. Consider now such (u',v') and (u,v). We prove that $\mathcal{R}$ satisfies the axioms of a weak refinement relation between (u',v') and (u,v):

1. Note that from $u' \,\mathcal{R}_1\, u$ (resp. $v' \,\mathcal{R}_2\, v$) it follows that $A_1' \subseteq A_1$ (resp. $A_2' \subseteq A_2'$) and further $A_1' \cap A_2 = A_1 \cap A_2' = \emptyset$. We have:

$$\begin{aligned}
V'((u',v'))\downarrow_A &= \left\{(Q_1 \cup Q_2)\downarrow_{A_1\cup A_2} \mid Q_1 \in V_1'(u'), Q_2 \in V_2'(v')\right\}\\
&= \{Q_1\downarrow_{A_1} \cup Q_2\downarrow_{A_2} \mid Q_1 \in V_1'(u'), Q_2 \in V_2'(v')\}\\
&\subseteq \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\} = V((u,v)) \ .
\end{aligned}$$

2. Let $z' \in [0,1]^{(k_1'k_2')}$ such that $\varphi'(u',v')(z')$. We now build the correspondence matrix Δ witnessing $(u',v')\,\mathcal{R}(u,v)$. Consider the correspondence matrices $\Delta^1 \in [0,1]^{k_1'\times k_1}$ and $\Delta^2 \in [0,1]^{k_2'\times k_2}$ witnessing respectively $u' \,\mathcal{R}_1\, u$ and $v' \,\mathcal{R}_2\, v$ for the transition vector z'. Define $\Delta = \Delta^1 \odot \Delta^2 \in [0,1]^{(k_1'k_2')\times(k_1k_2)}$. By Lemma 1, Δ is a correspondence matrix. Moreover, since $\varphi'(u',v')(z')$ holds, there exist $x' \in [0,1]^{k_1'}$ and $y' \in [0,1]^{k_2'}$ such that $\forall i,j,\ z'_{(i,j)} = x_i' y_j'$ and $\varphi_1'(u')(x')$ and $\varphi_2'(v')(y')$.

   - Let $(u'',v'') \in \{1,\dots,k_1'\}\times\{1,\dots,k_2'\}$ such that $z_{(u'',v'')} \neq 0$. By definition of x' and y', this implies that $x'_{u''} \neq 0$ and $y'_{v''} \neq 0$. Thus $\sum_{j=1}^{k_1} \Delta^1_{u''j} = 1$ and $\sum_{j=1}^{k_2} \Delta^2_{v''j} = 1$.

$$\begin{aligned}
\sum_{(r,s)\in\{1,\dots,k_1\}\times\{1,\dots,k_2\}} \Delta_{(u'',v'')(r,s)} &= \sum_{(r,s)\in\{1,\dots,k_1\}\times\{1,\dots,k_2\}} \Delta^1_{u''r}\Delta^2_{v''s}\\
&= \sum_{r=1}^{k_1}\sum_{s=1}^{k_2} \Delta^1_{u''r}\Delta^2_{v''s} = (\sum_{r=1}^{k_1} \Delta^1_{u''r})(\sum_{s=1}^{k_2} \Delta^2_{v''s}) = 1.
\end{aligned}$$

- Let $z = z'\Delta \in [0,1]^{(k_1 k_2)}$. Notice that $z = (x'\Delta^1) \otimes (y'\Delta^2)$.
  Let $x = x'\Delta^1$ and $y = y'\Delta^2$. Since $u' \mathcal{R}_1 u$ and $v' \mathcal{R}_2 v$, we have $\varphi_1(u)(x)$ and $\varphi_2(v)(y)$. Thus $\varphi(u,v)(z'\Delta)$.
- Let $u'', v'', u'''v'''$ such that $\Delta_{(u'',v'')(u''',v''')} \neq 0$. By definition, this implies that ${\Delta^1}_{u''u'''} \neq 0$ and ${\Delta^2}_{v''v'''} \neq 0$, and as a consequence $(u'', v'') \mathcal{R} (u''', v''')$.

We conclude that $\mathcal{R}$ is a weak refinement relation. Since $(o'_1, o'_2) \mathcal{R} (o_1, o_2)$, we have $S' \preceq S$. The second part of the theorem follows, as satisfaction is a special case of the refinement. □

As alphabets of composed CMCs have to be disjoint, the parallel composition cannot synchronize the components on state valuations like it is typically done for other (non-probabilistic) models. However, synchronization can be introduced by conjoining the parallel composition with a *synchronizer*—a single-state CMC whose valuation function relates the atomic propositions of the composed CMCs.

Example 6. *CMC $S \parallel S'$ of Figure 12b is synchronized with the synchronizer* Sync *given in Figure 12c.* Sync *removes from $S \parallel S'$ all the valuations that do not satisfy $(a = d) \wedge (b = \neg c)$. The result is given in Figure 12d. Observe that an inconsistency appears in State $(1,1)$. Indeed, there is no implementation of the two CMCs that can synchronize in the prescribed way. In general inconsistencies like this one can be uncovered by applying the pruning operator, which would return an empty specification. So synchronizers enable discovery of incompatibilities between component specifications in the same way as it is known for non-probabilistic specification models.*

Synchronization is associative with respect to parallel composition, which means that the order of synchronization and parallel composition is inessential for final functionality of the system.

Theorem 8. *Let S_1, S_2 and S_3 be three CMCs with pairwise disjoint sets of propositions A_1, A_2 and A_3. Let Sync_{123} be a synchronizer over $A_1 \cup A_2 \cup A_3$ and let Sync_{12} be the same synchronizer with its set of propositions restricted to $A_1 \cup A_2$. The following holds $[\![((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] = [\![(S_1 \parallel S_2 \parallel S_3) \wedge \mathsf{Sync}_{123}]\!]$.*

Proof. We first prove the following statement. Let S_1 and S_2 be two CMCs with disjoint sets of atomic propositions A_1 and A_2. Let Sync_1 be a synchronizing vector on A_1. We have $(S_1 \parallel S_2) \wedge \mathsf{Sync}_1 = (S_1 \wedge \mathsf{Sync}_1) \parallel S_2$.

First, remember that synchronizers are single state CMCs, with a single transition taken with probability 1. As a consequence, computing the conjunction with a synchronizer preserves the structure of any CMC. The only change lies in the sets of valuations.

Let p be a state of S_1 and q be a state of S_2. We have $(V_1(p) \cup V_2(q)) \cap V_{\mathsf{Sync}_1} \uparrow^{A_1 \cup A_2} = (V_1(p) \cap V_{\mathsf{Sync}_1}) \cup V_2(q)$. As a consequence, the valuations of $(S_1 \wedge \mathsf{Sync}_1) \parallel S_2$ are the same as the valuations of $(S_1 \parallel S_2) \wedge \mathsf{Sync}_1$.

By monotonicity of conjunction, we have $(S_1 \parallel S_2) \wedge \mathsf{Sync}_{12} \preceq (S_1 \parallel S_2)$. By Theorem 7, it is implied that $[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123}$, and finally $[\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] \subseteq [\![[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{S}_{123}]\!]$.

We now prove that $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}$. By monotonicity of conjunction, we have $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{12} \wedge \mathsf{Sync}_{123}$. Moreover, by the statement proved above, we have $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{12} \preceq ((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3$. As a consequence, we have $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}$, and thus $[\![[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] \subseteq [\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!]$.

□

Finally, synchronized parallel composition also supports component-based refinement in the style of Theorem 7.

Theorem 9. *If S_1', S_2', S_1, S_2 are CMCs, Sync is a synchronizer and $S_1' \preceq S_1$ and $S_2' \preceq S_2$, then $(S_1' \,\|\, S_2') \wedge \mathsf{Sync} \preceq (S_1 \,\|\, S_2) \wedge \mathsf{Sync}$.*

Consequently, a modeler can continue independent refinement of specifications under synchronization, knowing that the original synchronized specification will not be violated. The theorem is a direct corollary of precongruence (Theorem 7) and monotonicity of conjunction (follows from Theorem 6).

7.1 On comparing conjunction and parallel composition

We now compare conjunction and parallel composition with respect to implementation set inclusion. We shall see that if the two operations are defined on CMCs with independent sets of valuations, then parallel composition refines conjunction; the opposite does not hold. We first show that parallel composition refines conjunction.

Theorem 10. *Let S_1 and S_2 be consistent CMCs with $A_1 \cap A_2 = \emptyset$. It holds that $S_1 \,\|\, S_2 \preceq S_1 \wedge S_2$.*

Proof. Let $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be CMCs. Consider their parallel composition $S_1 \parallel S_2 = \langle \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}, (o_1, o_2), A, V^{\parallel} \rangle$ and their conjunction $S_1 \wedge S_2 = \langle \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}, (o_1, o_2), \varphi^{\wedge}, A, V^{\wedge} \rangle$, where $A = A_1 \cup A_2$. We build a refinement relation $\mathcal{R}$ on $(\{1, \ldots, k_1\} \times \{1, \ldots, k_2\}) \times (\{1, \ldots, k_1\} \times \{1, \ldots, k_2\})$ as $(u, v)\, \mathcal{R} (u', v')$ if and only if $u = u'$ and $v = v'$.

Let $(u, v) \in \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ such that $(u, v)\, \mathcal{R} (u, v)$. We now show that $\mathcal{R}$ is a refinement relation:

1. By construction, we have that $V^{\parallel}((u, v)) = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$. Moreover, since $A_1 \cap A_2 = \emptyset$, we have that $V^{\wedge}((u, v)) = V_1(u){\uparrow^A} \cap V_2(v){\uparrow^A} = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$. Thus $V^{\parallel}((u, v)) = V^{\wedge}((u, v))$.

2. Let $z = (z_{1,1}, z_{1,2}, \ldots, z_{k_1,k_2}) \in [0, 1]^{k_1 k_2}$ such that $\varphi^{\parallel}((u, v))(z)$ holds. Define the correspondence matrix $\Delta \in [0, 1]^{(k_1 k_2) \times (k_1 k_2)}$ as the matrix with $\Delta_{(u,v),(u,v)} = 1$ if $z_{u,v} \neq 0$ and 0 otherwise. Observe that $z\Delta = z$.

- Trivially, by construction, for all $(i,j) \in \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ such that $z_{i,j} \neq 0$, we have that $\sum_{i',j'} \Delta_{(i,j),(i',j')} = 1$.
- We prove that $\varphi^{\wedge}((u,v))(z)$ holds: By hypothesis, $\varphi^{\parallel}((u,v))(z)$ holds. So there exist $x \in [0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $\varphi_1(u)(x)$ holds, $\varphi_2(v)(y)$ holds and for all $i \in \{1,\dots,k_1\}$ and $j \in \{1,\dots,k_2\}$, we have $z_{i,j} = x_i y_j$. As a consequence, we have $\sum_{i=1}^{k_1} z_{i,j} = y_j$ for all $j \in \{1,\dots,k_2\}$ and $\sum_{j=1}^{k_2} z_{i,j} = x_i$ for all $i \in \{1,\dots,k_1\}$. Since both $\varphi_1(u)(x)$ and $\varphi_2(v)(y)$ hold, we have that $\varphi^{\wedge}((u,v))(z\Delta)$ holds.
- By construction of Δ, $\Delta_{(u,v),(u',v')} \neq 0$ implies that $u = u'$ and $v = v'$, and therefore implies $(u,v)\,\mathcal{R}(u',v')$.

We conclude that $\mathcal{R}$ is a refinement relation since $(o_1,o_2)\,\mathcal{R}(o_1,o_2)$. We have shown that $S_1 \parallel S_2 \preceq S_1 \wedge S_2$. □

A direct consequence of the above theorem is that any model of the parallel composition is a model for the conjunction, i.e.,$[\![S_1 \parallel S_2]\!] \subseteq [\![S_1 \wedge S_2]\!]$. We now show that the opposite inclusion does not hold.

Theorem 11. *Let S_1 and S_2 be consistent CMCs with $A_1 \cap A_2 = \emptyset$. It holds that $[\![S_1 \wedge S_2]\!] \not\subseteq [\![S_1 \parallel S_2]\!]$.*

Proof. We establish the proof by providing, in Figure 13, two CMCs S_1 and S_2 and a MC I, such that $I \models S_1 \wedge S_2$ and $I \not\models S_1 \parallel S_2$.

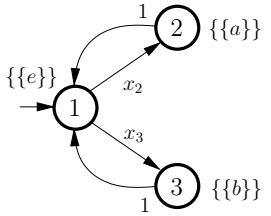
The common structure of conjunction and parallel composition is shown in Figure 13d. The constraint functions differ. According to the definitions of conjunction and parallel composition, we have $\varphi^{\wedge}(1,1)(z) \equiv z_{2,2} + z_{2,3} = z_{2,2} + z_{3,2} = 0.6 \wedge z_{3,2} + z_{3,3} = z_{2,3} + z_{3,3} = 0.4$ and $\varphi^{\parallel}(1,1)(z) \equiv z_{2,2} = 0.36 \wedge z_{2,3} = z_{3,2} = 0.24 \wedge z_{3,3} = 0.16$. I satisfies the conjunction, but not the parallel composition, since the probability mass 0.4 of going to state 2 in I cannot be distributed to $(2,2)$ of $S_1 \parallel S_2$. □

8 Disjunction and Universality

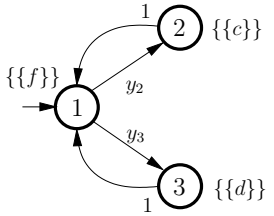
In this section we show that CMCs are not closed under disjunction. We then solve the *universality problem*, that is the problem of deciding whether a CMC admits arbitrary implementations.

8.1 On the Existence of a Disjunction of CMCs

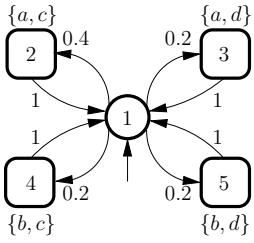
In this section we discuss the problem of computing a CMC S whose models are the union of the models accepted by two other CMCs $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$. In general, such a CMC may not exist. Indeed, assume that S_1 and S_2 have disjoint initial state valuations, and that the constraint functions of o_1 and o_2 do not share the same set of satisfying probability vectors. The initial state o of any specification representing the union could take valuations



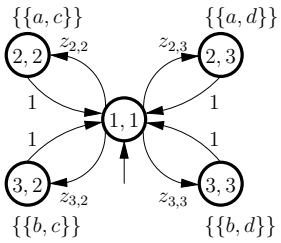
(a) S_1, $\varphi_1(1)(x) \equiv x_2 = .6 \wedge x_3 = .4$



(b) S_2, $\varphi_2(1)(y) \equiv y_2 = .6 \wedge y_3 = .4$



(c) I



(d) Common structure of $S_1 \wedge S_2$ and $S_1 \parallel S_2$.

Figure 13: Conjunction vs. parallel composition: $I \models S_1 \wedge S_2$ but $I \not\models S_1 \parallel S_2$.

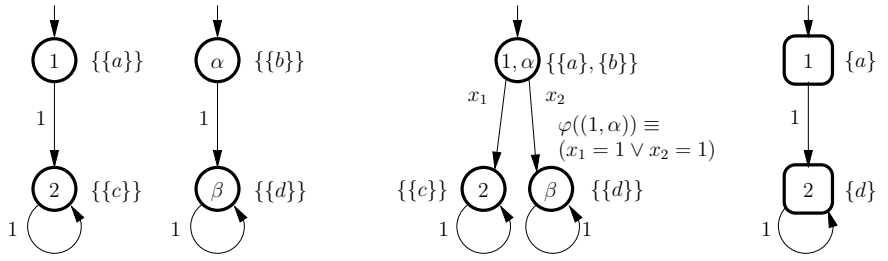
admissible according to o_1 and a distribution M_o according to o_2 (but not o_1). That is, we can not express the link between a choice of the valuation of the initial state and a probability distribution.

This is illustrated in Figure 14. S_1 admits implementations with valuation a in the initial state, and c afterwards. S_2 admits implementations with b in the initial state, and d afterwards. Any CMC representing the disjunction would have to admit a or b in the initial state, like the potential candidate S_3 in Figure 14c. Unfortunately, the implementation MC I in Figure 14d satisfies S_3 but it is a model of neither S_1 and S_2, as it combines the initial state valuation of the former, with behaviour of the latter.

However, if S_1 and S_2 have the same initial state valuation, then we can explicitly construct a CMC whose set of implementations is the union of the sets of implementations of S_1 and S_2. This CMC is called the *disjunction* of S_1 and S_2, and denoted $S_1 \vee S_2$.

Let $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be CMCs such that $V_1(o_1) = V_2(o_2)$. Then define $S_1 \vee S_2 = \langle Q, 0, \varphi, A, V \rangle$, where $Q = \{0, 1, \ldots, k_1, k_1 + 1, \ldots, k_1 + k_2\}$, $A = A_1 \cup A_2$ and

$$V(i) = \begin{cases} V_1(o) & \text{if } i = 0, o = o_1 = o_2 \\ V_1(i) & \text{if } i \in \{1, \ldots, k_1\} \\ V_2(i - k_1) & \text{if } i \in \{k_1 + 1, \ldots, k_1 + k_2\} \end{cases}$$



(a) CMC S_1

(b) CMC S_2

(c) CMC S_3 such that $[\![S_1]\!] \cup [\![S_2]\!] \subsetneq [\![S_3]\!]$

(d) MC I

Figure 14: CMCs not closed under disjunction: $I \models S_3$, but $I \not\models S_1$ and $I \not\models S_2$.

The constraint function $\varphi : Q \to [0,1]^{k_1+k_2+1} \to \{0,1\}$ is given by:

$$\varphi(i)(x_0, x_1, \ldots, x_{k_1}, x_{k_1+1}, \ldots, x_{k_1+k_2}) \equiv$$

$$\begin{cases} (\varphi_1(i)(x_1, \ldots, x_{k_1}) \vee \varphi_2(i)(x_{k_1+1}, \ldots, x_{k_1+k_2})) \wedge \sum_{i=0}^{k_1+k_2} x_i = 1 \\ \qquad\qquad \text{if i} = 0 \\ \varphi_1(i)(x_1, \ldots, x_{k_1}) \wedge (x_0 + \sum_{i=k_1+1}^{k_1+k_2} x_i = 0) \qquad \text{if } 1 \leq i \leq k_1 \\ \varphi_2(i)(x_{k_1+1}, \ldots, x_{k_1+k_2}) \wedge \sum_{i=0}^{k_1} x_i = 0 \qquad \text{if } k_1{+}1 \leq i \leq k_1{+}k_2 \end{cases}$$

8.2 The Universality Problem for CMCs

Consider the problem of deciding whether a CMC S admits all models defined over a set of atomic propositions A. This problem can be reduced to checking whether the set of implementations of the universal CMCs Univ^A representing all models over A is included in the set of implementations of S. The CMC Univ^A is defined as $\mathsf{Univ}^A = \langle \{1\}, 1, \varphi, A, V \rangle$, where $\varphi(1)(x) \equiv 1$ and $V(1) = 2^A$.

Theorem 12. *Let $\mathsf{Univ}^A = \langle \{1\}, 1, \varphi, A, V \rangle$ be the universal CMC on the set of atomic propositions A and let $I = \langle \{1, \ldots, n\}, o, M, A_I, V_I \rangle$ be any implementation such that $A \subseteq A_I$. We have that $I \models \mathsf{Univ}^A$.*

Proof. Construct the relation $\mathcal{R} = \{1, \ldots, n\} \times \{1\}$. We show that $\mathcal{R}$ is a satisfaction relation: Let $i \in \{1, \ldots, n\}$ such that $i \,\mathcal{R}\, 1$.

1. It is clear that $V_I(i)\downarrow_A \in V(1) = 2^A$.
2. Consider M_i. We build a correspondence matrix $\Delta \in [0,1]^{n \times 1}$ such that $\Delta_{j1} = 1$ if $M_{ij} > 0$, and 0 otherwise.

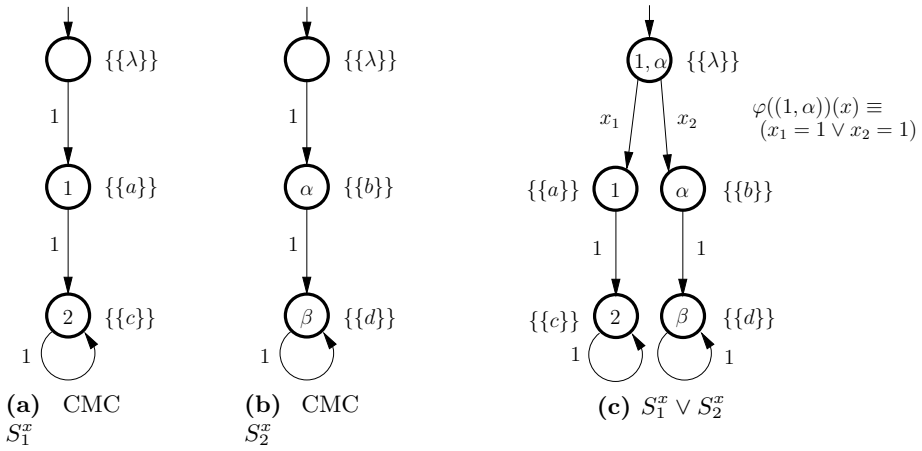


Figure 15: State-extended CMCs of Figure 14 and their disjunction.

- By construction, $\Delta_{j1} = 1$ for all j such that $M_{ij} > 0$.
- Since $M_i\Delta = 1$, $\varphi(1)(M_i\Delta)$ holds.
- Let i' such that $\Delta_{i',1} > 0$. By construction of $\mathcal{R}$, $i' \,\mathcal{R}\, 1$.

We conclude that $\mathcal{R}$ is a satisfaction relation, since $o \,\mathcal{R}\, 1$, and thus, $I \models \mathsf{Univ}^A$. □

We now switch to the problem of deciding whether the union of two CMCs S_1 and S_2 is universal. Despite the fact that CMCs are not closed under union, this problem has a relatively simple solution. The idea is to create a new initial state with a fresh atomic proposition $\lambda \notin A$ and then redistribute the entire probability mass to the original initial state. Formally:

Definition 11. *For a CMC $S = \langle\{1, \ldots, k\}, o, \varphi, A, V\rangle$ and an atomic proposition $\lambda \notin A$ define the state-extended CMC $S^x = \langle\{1, \ldots, k, o'\}, o', \varphi', A', V'\rangle$, where $A' = A \cup \{\lambda\}$, $V'(o') = \{\{\lambda\}\}$ and $V'(i) = V(i)$ for all $i \in \{1, \ldots, k\}$. The constraint function is given by:*

$$\varphi'(i)(x) \equiv \begin{cases} x_o = 1 \wedge (\sum_{i=1}^{k} x_i = 0) & \text{if } i = o' \\ \varphi(i)(x_1, \ldots, x_k) \wedge x_{o'} = 0 & \text{for } i \in \{1, \ldots, k\}. \end{cases}$$

Figure 15 gives an example. The union of the state-extended versions of S_1 and S_2 can now be computed and compared to the state-extended version of Univ^A. It is obvious that all the implementations of the state-extended version of a given CMC C are state-extended versions of implementations of C.

9 Deterministic CMCs

Clearly, if all implementations of a specification S_1 also implement a specification S_2, then the former is a strengthening of the latter. Indeed, S_1 specifies implementations that break no assumptions that can be made about implementations of S_2. Thus implementation set inclusion *is* a desirable refinement for specifications. Unfortunately the decision procedure for implementation set inclusion is more complex and less efficient than the weak and (in particular) the strong refinement. We have seen that the latter two both soundly approximate implementation set inclusion. Had that approximation been complete, we could have a more efficient and simpler to program procedure for implementing the implementation set inclusion.

In this section, we argue that this indeed is the case for an important subclass of specifications: *deterministic CMCs*. We show that for this class strong refinement coincides with the implementation set inclusion. Thus for deterministic CMCs more efficient algorithms exist for establishing the latter. For this reason we will also consider a determinization algorithm for CMCs.

A CMC S is *deterministic* iff for every state i, states reachable from i have pairwise disjoint admissible valuations:

Definition 12. *Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC. S is* deterministic *iff for all states $i, u, v \in \{1, \ldots, k\}$, if there exists $x \in [0,1]^k$ such that $(\varphi(i)(x) \wedge (x_u \neq 0))$ and $y \in [0,1]^k$ such that $(\varphi(i)(y) \wedge (y_v \neq 0))$, then we have that $V(u) \cap V(v) = \emptyset$ or $u = v$.*

By inspecting the constructions for conjunction and parallel composition, one can see that both trivially preserve determinism.

In Figures 3a and 3b, both S_1 and S_2 are deterministic specifications. In particular states 2 and 3, both of which can be reached from state 1 in both CMCs, have disjoint valuation sets. On the other hand, the CMC T given in Figure 16 is non-deterministic. Indeed, for States 2 and 3, which can both be reached from State 1, we have that $V_T(2) \cap V_T(3) = \{\{a, c\}\} \neq \emptyset$.

Deterministic CMCs are less expressive than non-deterministic ones, in the sense that the same implementation sets sometimes cannot be expressed. Consider again the CMC T given in Figure 16. The set of implementations of T cannot be represented by a deterministic CMC. Any merging of States 2 and 3 in T would result in a CMC that accepts models where one can loop on valuation $\{a, c\}$ and then accept valuation $\{a\}$ with probability 1. Such a model cannot be accepted by T.

We now present a determinization algorithm that can be applied to any CMC S in single valuation normal form (obtainable by applying the normalization algorithm of Definition 5). This algorithm is based on a subset construction that resembles the classical determinization for automata.

Definition 13. *Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a consistent CMC in single valuation normal form. If $Q \subseteq \{1, \ldots, k\}$ and $a \in 2^A$, then define $\mathsf{reach}(Q, a)$ to be the maximal*[1]

[1]Maximality is understood with respect to set inclusion here. It captures the fact that we want all such states.

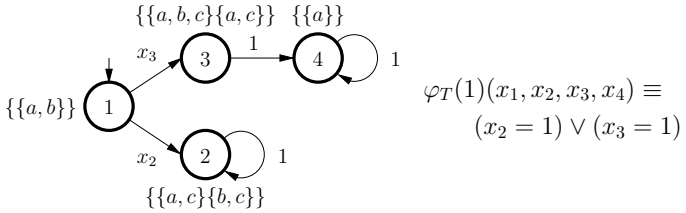


Figure 16: A deterministic CMC cannot express the implementation set of T.

set of states with valuation a that can be reached with a non-zero probability, using a distribution that satisfies the constraint of at least one state of Q. Formally $\mathsf{reach} : 2^{\{1,\dots,k\}} \times 2^A \mapsto 2^{\{1,\dots,k\}}$ *and*

$$\mathsf{reach}(Q,a) = \bigcup \{v \in \{1,\dots,k\} \mid V(v) = \{a\} \text{ and } \exists u \in Q.\, \exists x \in [0,1]^k.\, \varphi(u)(x) = 1 \wedge x_v > 0\}$$

We lift the notion of reachability to all possible valuations as follows:

$$\mathsf{Reach}(Q) = \{\mathsf{reach}(Q,a) \mid a \subseteq A\}$$

Now define the n-step reachability with

$$\mathsf{Reach}^n(Q) = \mathsf{Reach}^{n-1}(Q) \cup \bigcup_{Q' \in \mathsf{Reach}^{n-1}(Q)} \mathsf{Reach}(Q')$$

and its transitive closure as the fixpoint

$$\mathsf{Reach}^*(Q) = \{Q\} \cup \lim_{n\to\infty} \mathsf{Reach}^n(Q).$$

Observe that for all Q and for all $Q' \in \mathsf{Reach}^*(Q)$, there exists a valuation $a \in 2^A$ such that $V(q) = \{a\}$ for all $q \in Q'$ (by construction).

A deterministic CMC for S is the CMC $\rho(S) = \langle \{Q_1,\dots,Q_m\}, Q_{o'}, \varphi', A, V' \rangle$, where $\{Q_1,\dots,Q_m\} = \mathsf{Reach}^*(\{o\})$, $Q_{o'} = \{o\} \in \{Q_1,\dots,Q_m\}$ by definition, and φ' and V' are defined as follows:

- *for all $Q_i \in \{Q_1,\dots,Q_m\}$, let $V'(Q_i) = \{a\}$ iff for all $q \in Q_i$, $V(q_i) = \{a\}$ — there always exists exactly one such a by construction, and*
- *for all $Q_i \in \{Q_1,\dots,Q_m\}$ and for all $y \in [0,1]^m$,*

$$\varphi'(Q_i)(y_1,\dots,y_m) = [\forall 1 \le j \le m.\, Q_j \notin \mathsf{Reach}(Q_i) \implies y_j = 0] \wedge \bigvee_{q \in Q_i} \exists x \in [0,1]^k \Big[\varphi(q)(x) \wedge (\forall 1 \le j \le m.\; Q_j \in \mathsf{Reach}(Q_i) \implies \sum_{q' \in Q_j} x_{q'} = y_j) \Big]$$

Theorem 13. *Let S be a CMC in* single valuation normal form*. It holds that $S \preceq_S \rho(S)$.*

Proof. Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC in single valuation normal form. Let $\rho(S) = \langle \{Q_1, \ldots, Q_m\}, Q_{o'}, \varphi', A, V' \rangle$ be a determinization of S.

Define $\mathcal{R} \subseteq \{1, \ldots, k\} \times \{Q_1, \ldots, Q_m\}$ such that $u \,\mathcal{R}\, Q_i \iff u \in Q_i$. We will show that $\mathcal{R}$ is a strong refinement relation. Let u, i such that $u \,\mathcal{R}\, Q_i$.

1. By definition, since $u \in Q_i$, we have $V'(Q_i) = V(u)$.

2. Let $\Delta \in [0,1]^{k \times m}$ such that $\Delta_{vj} = 1$ if $Q_j \in \mathsf{Reach}(Q_i)$ and $v \in Q_j$, and 0 otherwise. We prove that Δ is a correspondence matrix.

   Suppose that there exist $1 \leq v \leq k$ and $1 \leq j \neq l \leq m$ such that $\Delta_{vj} = \Delta_{vl} = 1$. Then we know that $v \in Q_j$ and $v \in Q_l$, and that both Q_j and Q_l are in $\mathsf{Reach}(Q_i)$. This violates the definition of $\mathsf{Reach}(Q_i)$ (each of its sets must be maximal). As a consequence, for all $1 \leq v \leq k$, we have $\sum_{j=1}^{m} \Delta_{vj} \leq 1$ and Δ is a correspondence matrix.

   - Let $x \in [0,1]^k$ such that $\varphi(u)(x)$. Let $1 \leq v \leq k$ such that $x_v > 0$. Since $x_v > 0$ and $u \in Q_i$, we know that $\mathsf{Reach}(Q_i) \neq \emptyset$. Moreover, there exists $1 \leq j \leq m$ such that $Q_j \in \mathsf{Reach}(Q_i)$ and $v \in Q_j$. As a consequence, $\Delta_{vj} = 1$, and $\sum_{l=1}^{m} \Delta_{vl} = 1$.
   - Let $y = x\Delta$. We prove that $\varphi'(y)$ holds.
     - Since $\Delta_{vj} = 0$ for all $Q_j \notin \mathsf{Reach}(Q_i)$, we have that $y_j = 0$ for all $Q_j \notin \mathsf{Reach}(Q_i)$.
     - There exist $q \in Q_i$, namely u, and $x \in [0,1]^k$ defined above, such that $\varphi(u)(x)$ holds, and by definition, if $Q_j \in \mathsf{Reach}(Q_i)$, then for all $q' \in Q_j$, we have $\Delta_{q'j} = 1$. As a consequence, $y_j = \sum_{r=1}^{k} x_r \Delta_{rj} = \sum_{q' \in Q_j} x_{q'}$.

     Thus $\varphi'(x\Delta)$ holds.
   - If $\Delta_{vj} > 0$, then we have that $v \in Q_j$ by definition, thus $v \,\mathcal{R}\, Q_j$.

Finally, $\mathcal{R}$ is a strong refinement relation, and $o \in Q_{o'} = \{o\}$, thus S strongly refines $\rho(S)$.

□

This character of determinization resembles the known determinization algorithms for modal transition systems [97].

In Theorem 4 we have shown that weak (and thus also strong) refinement is sound with respect to implementation set inclusion. We now state one of the main theorems of the paper. Weak refinement is complete with respect to implementation set inclusion for deterministic CMCs:

Theorem 14. *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be two locally consistent deterministic CMCs with single admissible valuation in initial state and* $A_2 \subseteq A_1$*. We have* $[\![S_1]\!] \subseteq [\![S_2]\!] \Rightarrow S_1 \preceq S_2$*.*

Proof. First, since any consistent CMC with a single valuation in the initial state can be normalized (see Thm. 3), without change of the implementation set, we assume that S_1 and S_2 are actually in single valuation normal form.

Let $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be two consistent and deterministic CMCs in single valuation normal form such that $A_2 \subseteq A_1$ and $[\![S_1]\!] \subseteq [\![S_2]\!]$.

First, notice that $S_1 \preceq S_2 \iff S_1' = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_2, V_1{\downarrow_{A_2}} \rangle \preceq S_2$. It is thus safe to suppose that $A_1 = A_2 = A$. Similarly, if $I = \langle \ldots, A_I, V_I \rangle$ is a MC, we have $I \models S_1 \iff I' = \langle \ldots, A_1, V_I{\downarrow_A} \rangle \models S_1$. As a consequence, it is also safe to suppose that implementations have the same set of atomic propositions as S_1 and S_2.

In the following we will also rely on the local consistency of the two CMCs, which implies that for every state of a CMC there exists a MC satisfying it. Thanks to Theorem 2, local consistency is not a real limitation, as the specifications can always be pruned without loss of implementations.

In the proof we use the following notation: given a CMC S and its state o we write (S, o) to denote a new CMC created from S by assuming o as its initial state.

The proof is structured as a usual coinductive argument, starting with the presentation of a candidate relation $\mathcal{R}$ and then continuing with evidence that $\mathcal{R}$ is indeed a refinement relation witnessing the refinement of S_2 by S_1. The argument is essentially standard until the last step, marked with a $\star$ below, where we need to rely on the determinism of S_2 and an argument by contradiction to conclude.

Let $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ be the following binary relation on states:

$$\mathcal{R} = \{\, (v, u) \mid \text{For all } I.\ I \models (S_1, v) \text{ implies } I \models (S_2, u) \,\} \tag{4}$$

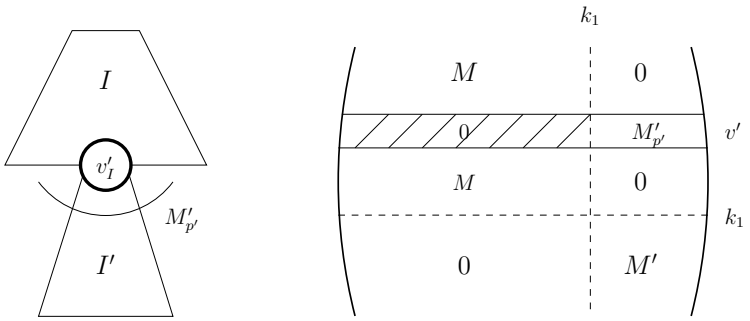
Consider v and u such that $v \,\mathcal{R}\, u$ and check that conditions of Def. 3 hold:

1. By local consistency of S_1 there exists a MC $I = \langle \{1, \ldots, n\}, p, M, A, V \rangle$ such that $I \models (S_1, v)$, and, since $v \,\mathcal{R}\, u$, then also $I \models (S_2, u)$. Thus $V(p) \in V_1(v)$ and $V(p) \in V_2(u)$. As S_1 and S_2 are in single valuation normal form, $V_1(v)$ and $V_2(u)$ are singletons, so $V_1(v) = V_2(u)$.

2. Consider a probability distribution vector $x \in [0, 1]^{k_1}$ such that $\varphi_1(v)(x)$. We want to build a correspondence matrix $\Delta \in [0, 1]^{k_1 \times k_2}$ such that $\varphi_2(u)(x\Delta)$ holds. To this end, we build another implementation MC $I = \langle \{1, \ldots, k_1\}, v, M, A, V \rangle$ such that for all $1 \leq w \leq k_1$,

   (i) $V(w)$ is the only valuation such that $V_1(w) = \{V(w)\}$; The existence of $V(w)$ is warranted by the normal form and local consistency of S_1.

   (ii) If $w \neq v$, the row M_w is any solution of $\varphi_1(w)$. One exists for each state w of S_1 because S_1 is locally consistent.



(a) The MC $\widehat{I}$ **(b)** The transition matrix $\widehat{M}$

Figure 17: Visualization of the construction of $\widehat{I}$ for the proof of Thm. 14.

(iii) $M_v = x$.

When necessary, we will address state w of I as w_I to differentiate it from state w of S_1. The MC I clearly satisfies (S_1, v) as witnessed by the identity satisfaction relation $\mathcal{R}_1$. By hypothesis, we thus have $I \models (S_2, u)$, as $v \,\mathcal{R}\, u$. Let $\mathcal{R}_2$ be the relation witnessing $I \models (S_2, u)$, and let Δ_2 be the correspondence matrix witnessing $v_I \,\mathcal{R}_2\, u$. We will now show the correspondence matrix Δ witnessing the weak refinement invariant for $v \,\mathcal{R}\, u$. Let $\Delta = \Delta_2$.

- $\forall 1 \leq i \leq k_1,\ x_i \neq 0 \Rightarrow \sum_{j=1}^{k_2} \Delta_{ij} = 1$, because $\Delta = \Delta_2$, which is a correspondence matrix witnessing satisfaction for the same probability vector $M_v = x$.
- $\varphi_2(u)(x\Delta)$ holds for the same reason.
- It remains to show that if $x_{v'} \neq 0$ and $\Delta_{v'u'} \neq 0$, then $v' \,\mathcal{R}\, u'$. This argument occupies the remainder of the proof.

$\star$ Assume v', u' as above. By definition of I and Δ we have that $(I, v'_I) \models (S_1, v')$ and $(I, v'_I) \models (S_2, u')$. We want to prove not only for (I, v'_I) but also for all implementations I' such that $I' \models (S_1, v')$ that $I' \models (S_2, u')$. This argument proceeds ad absurdum.

Suppose this is not the case: there exists a MC $I' = \langle \{1, \ldots, n\}, p', M', A, V' \rangle$ such that $I' \models (S_1, v')$ and $I' \not\models (S_2, u')$. Let $\mathcal{R}'$ be the satisfaction relation witnessing $I' \models (S_1, v')$. We will use this implementation to construct an implementation $\widehat{I}$ of (S_1, v) which also satisfies (S_2, u). Since $\widehat{I}$ will embed I', and S_2 is deterministic, we will be able to obtain that $I' \models (S_2, u')$, which is a contradiction. Thus I' cannot exist, and indeed $v' \,\mathcal{R}\, u'$.

Below we construct $\widehat{I}$ and argue that $\widehat{I} \models (S_1, v)$ and thus $\widehat{I} \models (S_2, u)$. In the last part of the proof, marked with a $\circledcirc$, we argue that $I' \models (S_2, u')$, which leads to a contradiction, concluding the proof.

Let $\widehat{I} = \langle \{1, \ldots, k_1, k_1+1, \ldots, k_1+n\}, v, \widehat{M}, A, \widehat{V} \rangle$, where, among others, v and n are defined above. Intuitively, the first k_1 states correspond to I and the next n states to I'. The state v'_I will be the link between the two and its outgoing transitions will be the ones of p', the state of I'. The construction is illustrated in Figure 17. The left part of the figure shows the general structure of $\widehat{I}$, the right part shows the composition of its transition matrix. Formally, we define the transition matrix $\widehat{M}$ as follows:

$$\widehat{M}_{ij} = \begin{cases} M_{ij} & \text{if } 1 \leq i,j \leq k_1 \text{ and } i \neq v' \\ 0 & \text{if } 1 \leq j \leq k_1 \text{ and } i = v' \\ 0 & \text{if } 1 \leq i \leq k_1 \text{ and } i \neq v',\ j > k_1 \\ M'_{p'(j-k_1)} & \text{if } j > k_1 \text{ and } i = v' \\ 0 & \text{if } i > k_1 \text{ and } 1 \leq j \leq k_1 \\ M'_{(i-k_1)(j-k_1)} & \text{if } i,j > k_1 \end{cases}$$

Furthermore:

$$\widehat{V}(i) = \begin{cases} V(i) & \text{for } i \leq k_1 \\ V'(i-k_1) & \text{for } i > k_1 \end{cases}$$

We want to show that $\widehat{I} \models (S_1, v)$ first. Consider the following relation $\widehat{\mathcal{R}} \subseteq \{1, \ldots, k_1+n\} \times \{1, \ldots, k_1\}$, between the states of $\widehat{I}$ and the states of S_1:

$$\widehat{\mathcal{R}} = \{(q,w) \in \mathcal{R}_1 \mid q \neq v'\} \cup \{(q,w) \mid (q-k_1)\,\mathcal{R}'\,w\} \cup \{(v',w) \mid p'\,\mathcal{R}'\,w\} \qquad (5)$$

Intuitively, $\widehat{\mathcal{R}}$ is equal to $\mathcal{R}_1$ for the states $q \leq k_1$, except v', and equal to $\mathcal{R}'$ for the states $q > k_1$. The states related to $v'_{\widehat{I}}$ are the ones that were related to p' with $\mathcal{R}'$. We will show that $\widehat{\mathcal{R}}$ is a satisfaction relation between $\widehat{I}$ and (S_1, v).

Let q, w be states of $\widehat{I}$ and S_1 respectively, such that $q\widehat{\mathcal{R}}w$. For all the pairs where $q \neq v'_{\widehat{I}}$, the conditions of the satisfaction relation still hold because they held for $\mathcal{R}_1$ if $q \leq k_1$ and for $\mathcal{R}'$ otherwise ($\mathcal{R}_1 \subseteq \widehat{\mathcal{R}}$ by construction, since $v'_{\widehat{I}}\widehat{\mathcal{R}}v'$ and moreover v' is the only state to which $v'_{\widehat{I}}$ was related in the identity relation $\mathcal{R}_1$). It remains to check the conditions for the pairs where $q = v'_{\widehat{I}}$, as this is the only state with a new behaviour with respect to I and I'. So consider a state w of S_1 such that $v'_{\widehat{I}}\widehat{\mathcal{R}}w$.

1. Because v'_I and $p'_{I'}$ are both related to v' (respectively in $\mathcal{R}_1$ and in $\mathcal{R}'$) it is clear that $\widehat{V}(v'_{\widehat{I}}) = \widehat{V}(p')$. As $p'\,\mathcal{R}'\,w$, we know that $V'(p') \in V_1(w)$. Thus, $\widehat{V}(v'_{\widehat{I}}) \in V_1(w)$.

2. Let the correspondence matrix Δ' witness $p'\,\mathcal{R}'\,w$. Let $\widehat{\Delta} \in [0,1]^{(k_1+n)\times k_1}$ such that $\widehat{\Delta}_{ij} = 0$ if $i \leq k_1$, and $\widehat{\Delta}_{ij} = \Delta'_{(i-k_1)j}$ otherwise.

   - We want to show that if $\widehat{M}_{(v'_{\widehat{I}})(w')} \neq 0$, then $\sum_{j=1}^{k_1} \widehat{\Delta}_{w'j} = 1$. We know that $\widehat{M}_{(v'_{\widehat{I}})(w')} = 0$ if $w' \leq k_1$. Take $w' > k_1$ such that $\widehat{M}_{(v'_{\widehat{I}})(w')} \neq 0$. Then we

know that $\widehat{M}_{(v'_{\widehat{I}})(w')} = M'_{p'(w'-k_1)}$. Because $\mathcal{R}'$ is a satisfaction relation, it implies that $\sum_{j=1}^{k_1} \Delta'_{(w'-k_1)j} = 1$. Thus, $\sum_{j=1}^{k_1} \widehat{\Delta}_{w'j} = \sum_{j=1}^{k_1} \Delta'_{(w'-k_1)j} = 1$.

- We want to show now that $\varphi_1(w)(\widehat{M}_{v'_{\widehat{I}}}\widehat{\Delta})$ holds. Let $1 \leq j \leq k_1$. We have:

$$\begin{aligned}\left[\widehat{M}_{v'_{\widehat{I}}}\widehat{\Delta}\right]_j &= \sum_{l=1}^{k_1+n} \widehat{M}_{(v'_{\widehat{I}})l}\widehat{\Delta}_{lj} = 0 + \sum_{l=k_1+1}^{k_1+n} \widehat{M}_{(v'_{\widehat{I}})l}\widehat{\Delta}_{lj} = \sum_{l=1}^{n} M'_{p'l}\Delta'_{lj} \\ &= \left[M'_{p'}\Delta'\right]_j .\end{aligned}$$

  As a consequence, $\widehat{M}_{v'_{\widehat{I}}}\widehat{\Delta} = M'_{p'}\Delta'$. Since Δ' is a witness of $p' \,\mathcal{R}'\, w$, $\varphi_1(w)(M'_{p'}\Delta')$ holds. So does $\varphi_1(w)(\widehat{M}_{v'_{\widehat{I}}}\widehat{\Delta})$.

- We want to show that if $\widehat{M}_{(v'_{\widehat{I}})q} \neq 0$ and $\widehat{\Delta}_{qw'} \neq 0$, then $q\widehat{\mathcal{R}}w'$. We only need to consider $q > k_1$ (since otherwise $\widehat{M}_{(v'_{\widehat{I}})q} = 0$) and w' such that $\widehat{\Delta}_{qw'} \neq 0$. In this case, $\widehat{M}_{(v'_{\widehat{I}})q} = M'_{p'(q-k_1)} \neq 0$ and $\Delta'_{(q-k_1)w'} \neq 0$. As Δ' is a witness of $p' \,\mathcal{R}'\, w$, it has to be that $(q-k_1)\,\mathcal{R}'\, w'$, which implies, by definition of $\widehat{\mathcal{R}}$, that $q\widehat{\mathcal{R}}w'$.

So we conclude that $\widehat{I} \models (S_1, v)$, and thus also $\widehat{I} \models (S_2, u)$ since $v \,\mathcal{R}\, u$.

◎ Finally, to reach the contradiction, we show that the above implies $I' \models (S_2, u')$. Since $\widehat{I} \models (S_2, u)$ there exists $\Delta'' \in [0,1]^{(k_1+n)\times k_2}$ such that $\varphi_2(u)(\widehat{M}_{v_{\widehat{I}}}\Delta'')$ holds.

(A) Consider $u'' \neq u'$ such that $V_2(u'') = V_2(u')$. Due to the determinism of S_2, and to the fact that u' is accessible from u, we have $[\widehat{M}_{v_{\widehat{I}}}\Delta'']_{u''} = 0$. Otherwise $\varphi_2(u)$ would be violated. Since $\widehat{M}_{(v_{\widehat{I}})(v'_{\widehat{I}})} \neq 0$ and $\widehat{M}_{(v_{\widehat{I}})(v'_{\widehat{I}})}\Delta''_{(v'_{\widehat{I}})u''}$ is part of $[\widehat{M}_{v_{\widehat{I}}}\Delta'']_{u''}$, we must have $\Delta''_{(v'_{\widehat{I}})u''} = 0$.

(B) Consider u''' such that $V(u''') \neq V(u')$. It is clear that $\Delta''_{(v'_{\widehat{I}})u'''} = 0$ since Δ'' is witnessing satisfaction between $\widehat{I}$ and S_2.

(C) Moreover, we know that $\widehat{M}_{(v_{\widehat{I}})(v'_{\widehat{I}})} \neq 0$. Thus, $\sum_{j=1}^{k_2} \Delta''_{v'_{\widehat{I}}j} = 1$.

According to (A) and (B), the only non-zero value in the sum in (C) must be $\Delta''_{(v'_{\widehat{I}})u'}$. Since Δ'' is witnessing $\widehat{I} \models (S_2, u)$, we have $(\widehat{I}, v'_{\widehat{I}}) \models (S_2, u')$. But $v'_{\widehat{I}}$ and p' only differ by state names, not by behaviours, so $(I', p') \models (S_2, u')$. This contradicts our assumption. Thus $v' \,\mathcal{R}\, u'$, and $\mathcal{R}$ is a weak refinement relation.

Finally, the hypothesis $[\![S_1]\!] \subseteq [\![S_2]\!]$ implies that $o_1 \,\mathcal{R}\, o_2$ and further $S_1 \preceq S_2$. □

Thus, weak refinement and the implementation set inclusion coincide on the class of deterministic CMCs with at most one valuation in the initial state. Finally, Theorem 14 also holds for strong refinement, as the two refinements coincide for deterministic CMCs. Before any formal introduction of the result, we first introduce the following lemma that characterizes the shape of the witness matrix of the satisfaction relation for an implementation and a CMC in normal form.

Lemma 15. *Let $S = \langle \{1, \dots, k\}, o_S, \varphi, A, V_S \rangle$ be a deterministic CMC in single valuation normal form. Let $P = \langle \{1, \dots, n\}, o_P, M, A, V_P \rangle \in [\![S]\!]$ witnessed by a satisfaction relation $\mathcal{R}$. Let $p \in \{1, \dots, n\}$ and $u \in \{1, \dots, k\}$ such that $p \mathcal{R} u$, and let Δ be the associated correspondence matrix. We have*

$$\forall p' \in \{1, \dots, n\}.\ M_{pp'} \neq 0 \Rightarrow \left|\left\{u' \in \{1, \dots, k\} \mid \Delta_{p'u'} \neq 0\right\}\right| = 1.$$

Proof. Let $S = \langle \{1, \dots, k\}, o_S, \varphi, A, V_S \rangle$ be a deterministic CMC in single valuation normal form. Let $P = \langle \{1, \dots, n\}, o_P, M, A, V_P \rangle \in [\![S]\!]$ witnessed by a satisfaction relation $\mathcal{R}$. Let $p \in \{1, \dots, n\}$ and $u \in \{1, \dots, k\}$ such that $p \mathcal{R} u$, and let Δ be the associated correspondence matrix.

Suppose that there exist p', u' and u'' such that $M_{pp'} > 0$, $\Delta_{p'u'} > 0$ and $\Delta_{p'u''} > 0$ with $u' \neq u''$. Let $y = M\Delta$ be the probabilistic transition outgoing from u according to Δ. By construction, we have $y_{u'} > 0$ and $y_{u''} > 0$.

Moreover, since $\Delta_{p'u'} > 0$ and $\Delta_{p'u''} > 0$, it holds that $p' \mathcal{R} u'$ and $p' \mathcal{R} u''$. Because of the single valuation normal form of S, this implies that $V_S(u') = V_S(u'') = \{V_P(p')\}$.

Finally, there exist u, u' and $u'' \in \{1, \dots, k\}$ with $u' \neq u''$ and $y \in [0, 1]^k$ such that $\varphi(u)(y) = 1$, $y_{u'} > 0$, $y_{u''} > 0$ and $V_S(u') = V_S(u'')$. This breaks the assumption that S is deterministic, which concludes the proof.

□

According to the lemma, in any MC implementing a deterministic CMC, the probability of going to one implementation state is never distributed to more than one specification state. Otherwise the specification would be non-deterministic. We are now ready to state the theorem.

Theorem 16. *Let $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A, V_1 \rangle$ and $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A, V_2 \rangle$ be two deterministic CMCs in normal form. If there exists a weak refinement relation $\mathcal{R}$ such that $S_1 \mathcal{R} S_2$, then $\mathcal{R}$ is also a strong refinement relation.*

Proof. Let $v \in \{1, \dots, k_1\}$ and $u \in \{1, \dots, k_2\}$ such that $v \mathcal{R} u$.

1. By hypothesis, $V_1(v) \subseteq V_2(u)$;
2. We know that for all $x \in [0, 1]^{k_1}$ satisfying $\varphi_1(v)$, there exists a correspondence matrix Δ^x satisfying the axioms of weak refinement. We will build a correspondence matrix Δ^0 that will work for all x. Let $p \in \{1, \dots, k_1\}$.

   If for all $x \in [0, 1]^{k_1}$, $\varphi_1(v)(x) \Rightarrow x_p = 0$, then let $\Delta^0_p = (0, \dots, 0)$.

Else, consider $x \in [0,1]^{k_1}$ such that $\varphi_1(v)(x)$ and $x_p \neq 0$. By hypothesis, there exists a correspondence matrix Δ^x associated to $v \mathcal{R} u$. Let $\Delta^0_p = \Delta^x_p$. By Lemma 15, there is a single $u' \in \{1, \ldots, k_2\}$ such that $\Delta^x_{pu'} \neq 0$. Moreover, by definition of Δ^x, we know that $\sum_{r=1}^{k_2} \Delta^x_{pr} = 1$, thus $\Delta^x_{pu'} = 1$. So Δ^0 is a correspondence matrix.

Suppose there exists $y \neq x \in [0,1]^{k_1}$ such that $\varphi_1(v)(y)$ and $y_p \neq 0$. Let Δ^y be the associated correspondence matrix. As for x, there exists a unique $u'' \in \{1, \ldots, k_2\}$ such that $\Delta^y_{pu''} \neq 0$. Moreover $\Delta^y_{pu''} = 1$. Let $x' = x\Delta^x$ and $y' = y\Delta^y$. By definition, both $\varphi_2(u)(x')$ and $\varphi_2(u')(y')$ hold, $x'_{u'} \neq 0$ and $y'_{u''} \neq 0$. As $\Delta^x_{pu'} = \Delta^y_{pu''} = 1$, we have $V_2(u') \cap V_2(u'') \neq \emptyset$. By hypothesis, S_2 is deterministic, thus $u' = u''$.

As a consequence, we have $\Delta^x_p = \Delta^y_p$, so $\forall z \in [0,1]^{k_1},\ (\varphi_1(v)(z) \wedge (z_p \neq 0)) \Rightarrow \Delta^z_p = \Delta^0_p$, and we conclude that Δ^0 is uniquely defined.

Finally, consider Δ^0 defined as above. Let $x \in [0,1]^{k_1}$ such that $\varphi_1(v)(x)$:

- $x_i \neq 0 \Rightarrow \Delta^0_i = \Delta^x_i \Rightarrow \sum_{j=1}^{k_2} \Delta^0_{ij} = 1$;
- $x\Delta^0 = x\Delta^x$, thus $\varphi_2(u)(x\Delta^0)$ holds;
- If $\Delta^0_{v'u'} \neq 0$, then there exists $y \in [0,1]^{k_1}$ such that $\varphi_1(v)(y)$ and $\Delta^0_{v'u'} = \Delta^y_{v'u'}$, thus $v' \mathcal{R} u'$.

We conclude that $\mathcal{R}$ is a strong refinement relation. □

Let us summarize the relations between refinements for deterministic CMCs:

$$[\![S_1]\!] \subseteq [\![S_2]\!] \quad \text{iff} \quad S_1 \preceq S_2 \quad \text{iff} \quad S_1 \text{ strongly refines } S_2 \ , \tag{6}$$

if the CMCs are consistent and have at most one valuation in the initial state. The coincidence of the semantic refinement with coinductive refinements for CMCs is analogous to the case of trace inclusion refinement and simulation for deterministic transition systems.

The above results on completeness for deterministic specifications carry over to refinements of [46] and [48] for IMCs, which are special cases of our refinements. Completeness of these refinements was an open problem until now.

Discussion: A Weaker Notion of Determinism Our notion of determinism may appear overly strong. Indeed, it assumes that, from a given state i, one cannot reach two states u and v that share common sets of valuations. The assumption is made independently of the distributions used to reach the two states, i.e., it may be the case that there exists no distribution through which both u and v can be reached simultaneously.

A natural way to solve the problem would be to consider a weaker version of determinism. More precisely, we say that a CMC $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \}$ is weakly

deterministic if whenever there exists $x \in [0,1]^k$ and states i, u, v such that $\varphi(i)(x)$ and $x_u > 0$ and $x_v > 0$, we have $V(u) \cap V(v) = \emptyset$. This version of determinism is strictly weaker than the one given in Definition 12. Indeed, only states that can be reached by the same distribution should have disjoint sets of valuations. Though this notion seems reasonable, as mentioned, one can show that there exist two weakly deterministic CMCs S_c and S_d such that $[\![S_c]\!] \subseteq [\![S_d]\!]$ but $S_c \not\preceq S_d$. Example of such CMCs are given in the second item of Proposition 5 on page 87. Hence working with this weaker, even if more natural, version of determinism does not close the gap between weak refinement and implementation set inclusion.

10 Polynomial CMCs

It is not surprising that CMCs are closed under both conjunction and parallel composition. Indeed, CMCs do not make any assumptions on constraint functions, even though there are many classes of constraints that are practically intractable. While this paper is mainly concerned with the development of the theoretical foundations for CMCs, we now briefly study classes of CMCs for which operations on constraints required by our algorithms can be managed quite efficiently.

A first candidate could be linear constraints, which is the obvious generalization of interval constraints. Unfortunately, linear constraint CMCs are not closed under parallel composition. Indeed, as we have seen in Section 7 the parallel composition of two linear constraints leads to a polynomial constraint. However, what is more interesting is that polynomial constraints *are* closed under both conjunction and parallel composition and that these operations do not increase the quantifier alternations since they only introduce existential quantifiers. Hence, one can claim that CMCs with polynomial constraints and only existential quantifiers are certainly the smallest extension of IMCs closed under all operations.

From the algorithmic point of view, working with polynomial constraints should not be seen as an obstacle. First, we observe that algorithms for conjunction and parallel composition do not require any complex operations on polynomials. The refinement algorithms (presented in Section 5.2) are polynomial in the number of states, and each iteration requires a quantifier elimination. This procedure is known to be double exponential in general, but there exist efficient single exponential algorithms [93, 94] when quantifier alternations are fixed. Those algorithms are implemented in Maple [95]. The pruning operation is polynomial in the number of states, but each iteration also requires an exponential treatment as one has to decide whether the constraints have at least one solution. Again, such problems can be solved with efficient algorithms. Finally, determinizing a CMC can be performed with a procedure that is similar to the determinization procedure for finite-state automata. Such a procedure is naturally exponential in the number of states.

Remark 2. *In Section 7, it was shown that, assuming independent sets of valuations, parallel composition is refined by conjunction. We have also observed that the conjunction or disjunction of two linear constraints remains linear, but that parallel composition*

may introduce polynomial constraints. From an implementation point of view it may thus be more efficient to work with linear constraints only. For doing so, one can simply approximate parallel composition with conjunction.

11 Relating CMCs to Probabilistic Automata

CMCs are a newcomer in a long series of probabilistic modeling languages and abstractions for them. Throughout the paper we have indicated that many of our results directly translate to simpler abstractions, like IMCs. We shall now further discuss this foundational aspect of CMCs, showing how they subsume a few established notions of refinement and parallel composition for probabilistic automata (and for process algebra based on them).

Below we write $\mathrm{Dist}(S)$ for the set of all probability distributions over a finite set S. Given two finite sets S and T and a probability distribution $\alpha \in \mathrm{Dist}(S \times T)$, we denote the marginal distribution over S as $\alpha_{s,T} = \sum_{t \in T} \alpha_{s,t}$, and similarly for T. We say that φ is a *non-deterministic distribution constraint* over set I if all solutions x of φ are point distributions; so $\exists i.\, x_i = 1$. Write $\left[\begin{smallmatrix} i \\ S \end{smallmatrix}\right]$ to denote a particular point distribution for which $\left[\begin{smallmatrix} i \\ S \end{smallmatrix}\right]_i = 1$. For example, in Figure 11 constraints $\varphi_c(2)$ and $\varphi_d(1)$ are non-deterministic distribution constraints. The two point distributions satisfying $\varphi_c(2)$ are $\left[\begin{smallmatrix} 3 \\ 1..4 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 4 \\ 1..4 \end{smallmatrix}\right]$.

Non-deterministic distribution constraints model a non-deterministic choice of an element from S. They will be used to encode non-determinism in CMCs.

A probabilistic automaton (PA for short) [6] is a tuple $\mathbb{S} = (S, Act, \rightarrow, s_1)$, where S is a finite set of states, $\rightarrow \subseteq S \times Act \times \mathrm{Dist}(S)$ is a finite transition relation and $s_1 \in S$ is the initial state.

If $\pi \in \mathrm{Dist}(S)$ and $\rho \in \mathrm{Dist}(T)$, then $\pi \otimes \rho$ denotes the unique independent product distribution such that $(\pi \otimes \rho)_{st} = \pi_s \rho_t$. This is consistent with our definition of $\otimes$ in Section 3, if π, ρ and $\pi \otimes \rho$ are interpreted as row vectors. Then the *derived combined transition relation* of $\mathbb{S}$ is given by $\rightarrow_c \in S \times Act \times \mathrm{Dist}(S)$ as follows. We say that $t \xrightarrow{a}_c \rho$ iff ρ is a convex linear combination of vectors from $\boldsymbol{\rho} = \{\rho_i \mid t \xrightarrow{a} \rho_i\}$, so $\rho = \boldsymbol{\rho} \times \lambda$, where λ is a distribution vector $\lambda \in [0,1]^{|\boldsymbol{\rho}|}$. We interpret $\boldsymbol{\rho}$ as a matrix, where ith column is a distribution ρ_i.

Consider two PA $\mathbb{S} = (S, Act, \rightarrow^S, s_0)$ and $\mathbb{T} = (T, Act, \rightarrow^T, t_0)$. For a binary relation $R \subseteq S \times T$ we define a derived relation $R^* \subseteq \mathrm{Dist}(S) \times \mathrm{Dist}(T)$ such that $\pi R^* \rho$ iff there exists a distribution $\alpha \in \mathrm{Dist}(S \times T)$ and (1) $\alpha_{q,T} = \pi_q$ for all $q \in S$, (2) $\alpha_{S,r} = \rho_r$ for all $r \in T$ and (3) $\alpha_{s,t} \neq 0$ implies sRt.

Definition 14 (Simulation [6])**.** *A relation $R \subseteq S \times T$ is a* simulation *iff $(s,t) \in R$ implies that whenever $s \xrightarrow{a} \pi$ for a distribution π, then $t \xrightarrow{a} \rho$ for distribution ρ such that $\pi R^* \rho$.*

R is a probabilistic *simulation iff $(s,t) \in R$ implies that if $s \xrightarrow{a} \pi$, then $t \xrightarrow{a}_c \rho$ for some distribution ρ, and $\pi R^* \rho$.*

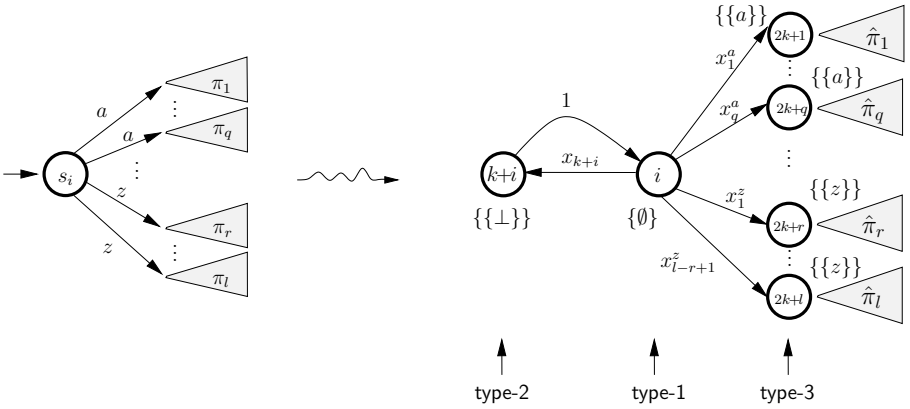


Figure 18: Reducing a PA to CMC. There $\widehat{\pi}$ denotes a distribution constraint, which has a unique solution π. This is formalized below as $\widehat{\varphi}(2k+i')(x)$.

Let $A \subseteq Act$ be the subset of actions on which $\mathbb{S}$ and $\mathbb{T}$ should synchronize. The *parallel composition* of $\mathbb{S}$ and $\mathbb{T}$ is a PA $\mathbb{S} \parallel \mathbb{T} = (S \times T, Act, \to, (s_0, t_0))$, where $\to$ is the largest transition relation such that $(s,t) \xrightarrow{a} \pi \otimes \rho$ if:

$a \in A$ and $s \xrightarrow{a}{}^S \pi$ and $t \xrightarrow{a}{}^T \rho$, or
$a \notin A$ and $s \xrightarrow{a}{}^S \pi$ and $\rho = [{}^t_T]$, or
$a \notin A$ and $\pi = [{}^s_S]$ and $t \xrightarrow{a}{}^T \rho$.

We now propose a linear encoding of PAs into CMCs, which reduces simulation and parallel composition of PAs to refinement and parallel composition of CMCs (see Figure 18). Let $\mathbb{S} = (\{s_1, \ldots, s_k\}, Act, \to, s_0)$ be a PA. And let l be the number of reachable action-distribution pairs, so $\Omega_{\mathbb{S}} = \{(a_1, \pi_1), \ldots, (a_l, \pi_l)\} = \{(a, \pi) \mid \exists s \in S.\, s \xrightarrow{a} \pi\}$. The corresponding CMC is

$$\widehat{\mathbb{S}} = \langle \{1, \ldots, 2k{+}l\}, 1, \widehat{\varphi}, \mathrm{Act} \cup \bot, \widehat{V} \rangle \text{ , where } \bot \notin Act \text{ .}$$

$\widehat{\mathbb{S}}$ has three kinds of states. Type-1 states, $1 \ldots k$, correspond directly to states of $\mathbb{S}$. Distributions leaving these states model a non-deterministic choice. Type-2 states, $k{+}1, \ldots, 2k$, model a possibility that a component remains idle in a state. Type-3 states, $2k{+}1, \ldots, 2k{+}l$ model the actual distributions of $\mathbb{S}$. In the following we use i to range over states of Type-1 (so usually $1 \le i \le k$) and i' to range over action–distribution pairs (so usually $1 \le i' \le l$). Similarly for j.

$\widehat{V}$ assigns value $\{\emptyset\}$ to type-1 states and value $\{\{\bot\}\}$ to type-2 states. For type-3 states we assign actions of transitions in $\mathbb{S}$: $\widehat{V}(2k+i') = \{\{a_{i'}\}\}$ for $1 \le i' \le l$. The distribution constraints are as follows:

$\widehat{\varphi}(i)(x)$ iff i is type-1 and $x = [{}^{k+i}_{1..2k+l}]$ or $s_i \xrightarrow{a_{i'}} \pi_{i'} \wedge x = [{}^{2k+i'}_{1..2k+l}]$ for $1 \le i' \le l$.
$\widehat{\varphi}(k+i)(x)$ iff $k+i$ is type-2 and $x = [{}^{i}_{1..2k+l}]$.
$\widehat{\varphi}(2k+i')(x)$ iff $2k+i'$ is type-3 and $\forall j \in \{1, \ldots, k\}.\, x_j = \pi_{i'}(s_j)$

We can now relate simulation of PAs to refinement of CMCs. We say that a constraint is a single-point constraint if it is only satisfied by a unique distribution. Observe that all constraints in the encoding presented above are non-deterministic distribution constraints or single-point constraints.

Lemma 17. *Let φ and ψ be single-point constraints. If for each $x \in [0,1]^{k_1}$ such that $\varphi(x)$ holds, there exists a correspondence matrix $\Delta_x \in [0,1]^{k_1 \times k_2}$ such that $\psi(x\Delta_x)$ holds, then there exists a correspondence matrix $\Delta \in [0,1]^{k_1 \times k_2}$ such that for all $x \in [0,1]^{k_1}$ we have that $\varphi(x) \implies \psi(x\Delta)$.*

The lemma holds trivially because there is only one distribution satisfying φ.

Lemma 18. *Let φ (respectively ψ) be a non-deterministic distribution constraint over $\{1,\dots,k_1\}$ (respectively $\{1,\dots,k_2\}$). Then if for each distribution vector x satisfying φ there exists a correspondence matrix $\Delta_x \in [0,1]^{k_1 \times k_2}$ such that $\psi(x\Delta_x)$ holds, then there exists a correspondence matrix $\Delta \in [0,1]^{k_1 \times k_2}$ such that for all $x \in [0,1]^{k_1}$ we have that $\varphi(x) \implies \psi(x\Delta)$.*

Proof. Let x be such that $\varphi(x)$ holds (thus there exists $1 \leq i \leq k_1$ such that $x_i = 1$). There is a finite number of such vectors. Let x^i denote the one that has 1 on the ith position. Take Δ such that $\Delta_i = (\Delta_{x^i})_i$ (the witness from the lemma assumption) if x^i satisfies φ and $\Delta_i = 0^{k_2}$ otherwise.

Now for each x^i satisfying φ we have that $x^i\Delta = x^i\Delta_{x^i}$ and then $\varphi(x^i) \implies \psi(x^i\Delta_{x^i}) \iff \psi(x^i\Delta)$. □

Corollary 19. *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ we have that $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$ iff $\widehat{\mathbb{S}}$ weakly refines $\widehat{\mathbb{T}}$.*

Lemma 20. *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\mathbb{T}$ simulates $\mathbb{S}$ we have that $\widehat{\mathbb{S}}$ weakly refines $\widehat{\mathbb{T}}$.*

Proof. (sketch) Let $\mathcal{R} \subseteq S \times T$ be the relation witnessing the simulation of $\mathbb{S}$ by $\mathbb{T}$. Consider a relation $\mathcal{Q}$ as follows:

$$
\begin{aligned}
\mathcal{Q}_1 &= \{(i,j) \mid i \in \{1,\dots,k_1\}, j \in \{1,\dots,k_2\}, (s_i,t_j) \in \mathcal{R}\} \\
\mathcal{Q}_2 &= \{(k_1+i, k_2+j) \mid i \in \{1,\dots,k_1\}, j \in \{1,\dots,k_2\}, (s_{i-k_1}, t_{j-k_2}) \in \mathcal{R}\} \\
\mathcal{Q}_3 &= \{(2k_1+i', 2k_2+j') \mid i' \in \{1,\dots,l_1\}, j' \in \{1,\dots,l_2\}, (a_i,\pi_i) \in \Omega_{\mathbb{S}}, \\
&\qquad (a_j,\rho_j) \in \Omega_{\mathbb{T}}, a_i = a_j, (\pi_i,\rho_i) \in \mathcal{R}^*\} \\
\mathcal{Q} &= \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \mathcal{Q}_3
\end{aligned}
$$

It is easy to show that $\mathcal{Q}$ is a weak refinement. First observe that valuations always match for pairs in $\mathcal{Q}$. The valuation is empty for both S and T in $\mathcal{Q}_1$, it is $\{\bot\}$ in $\mathcal{Q}_2$, and $\{a_i\}$ in $\mathcal{Q}_3$.

For a pair in $(i,j) \in \mathcal{Q}_1$ a distribution vector x satisfying the constraint of S is always a point distribution. If $x_{k_1+i} = 1$, take $\Delta_{k_1+i,k_2+j} = 1$ and zero otherwise. If

$x_{2k_1+i'} = 1$ take $\Delta_{2k_1+i',2k_2+j'} = 1$ and zero otherwise, where j' is such that $t_{j'} \xrightarrow{a_{i'}} \rho_{j'}$ and $\pi_{i'} R^* \rho_{j'}$.

For a pair $(k_1 + i, k_2 + j) \in Q_2$ take $\Delta_{ij} = 1$, and zero otherwise.

For a pair $(2k_1 + i', 2k_2 + j') \in Q_3$ take Δ such that for $(i, j) \in \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ we have $\Delta_{ij} = \alpha_{ij}/x_i$, or zero if $x_i = 0$, where α is the distribution witnessing $\pi_{i'} R^* \rho_{j'}$. □

Lemma 21. *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$ we have that $\mathbb{T}$ simulates $\mathbb{S}$.*

Proof. (sketch) Assume that $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$ is witnessed by a relation $R \subseteq \{1, \ldots, 2k_1 + l_1\} \times \{1, \ldots, 2k_2 + l_2\}$. Show that a relation $\mathcal{Q} = \{(s_i, t_j) \in S \times T \mid (i, j) \in R, i \in \{1, \ldots, k_1\}, j \in \{1, \ldots, k_2\}\}$ is a simulation relation.

In the crucial point of the proof consider $\alpha_{s_i t_j} = \Delta_{ij} \pi_{i'}(s_i)$, where $\pi_{i'}$ is a distribution being the only solution of a point constraint for state $i' \in \{2k_1, \ldots, 2k_2 + l_1\}$. □

Theorem 22 follows as a corollary from the above two lemmas and Corollary 19.

Theorem 22. *$\mathbb{T}$ simulates $\mathbb{S}$ iff $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$.*

The same encoding is used to characterize parallel composition of PAs using parallel composition of CMCs.

We say that two CMCs S_1 and S_2 are isomorphic if there exists a bijection $f : \{1, \ldots, k_1\} \to \{1, \ldots, k_2\}$, such that $\varphi(v)$ is satisfied by $x \in [0, 1]^{k_1}$ if and only if $\varphi(f(v))$ is satisfied by x.

Expression $S[{}^{a'_1}/_{a_1}; \ldots; {}^{a'_n}/_{a_n}]_{a_1,\ldots,a_n \in Act}$ denotes a comprehended substitution, substituting a primed version of name a_i for each occurrence in a_i, for all actions in *Act*.

Theorem 23. *For two PAs $\mathbb{S}$ and $\mathbb{T}$ over the same set of actions Act and a synchronizing set $A \subseteq Act$ we have that $\widehat{\mathbb{S} \parallel \mathbb{T}}$ is isomorphic to*

$$((\widehat{\mathbb{S}} \parallel \widehat{\mathbb{T}}[{}^{a'}/_{a}]_{a \in Act \cup \bot}) \wedge \mathsf{S}_A)\, [{}^{a}/_{(a,a')};\ {}^{a}/_{(a,\bot')};\ {}^{a}/_{(\bot,a')}]_{a \in Act},$$

where S_A is a synchronizer over $Act \cup \bot \times Act' \cup \bot'$ defined by

$$(\forall a \in A.\, a \iff a') \wedge (\forall a \notin A.\, (a \Longrightarrow \bot') \wedge (a' \Longrightarrow \bot))$$

Proof. Let $\mathbb{S} = (\{s_1, \ldots, s_{k_1}\}, Act, \to^S, s_1)$, $\mathbb{T} = (\{t_1, \ldots, t_{k_2}\}, Act, \to^T, t_1)$, and $A \subseteq Act$. Consider $\mathbb{S} \parallel \mathbb{T} = (\{(s_1, t_1), (s_1, t_2), \ldots, (s_{k_1}, t_{k_2})\}, Act, \to, (s_1, t_1))$ defined in the usual way.

We now construct $\widehat{\mathbb{S} \parallel \mathbb{T}} = (\{1, \ldots, 2k_1k_2 + l\}, 1, \widehat{\varphi}, Act \cup \bot, \widehat{V}\})$ in the usual way, where l is the number of reachable action-distribution pairs.

Consider $\widehat{\mathbb{S}} \parallel \widehat{\mathbb{T}}[{}^{a'}/_{a}]_{a \in Act \cup \bot} = \langle \{1, \ldots, 2k_1 + l_1\} \times \{1, \ldots, 2k_2 + l_2\}, (1, 1), \varphi, Act \cup Act' \cup \bot \cup \bot', V \rangle$, where l_1 and l_2 are the number of reachable action-distribution pairs for $\mathbb{S}$ and $\mathbb{T}$, respectively. Conjoining with S_A allows exactly those pairs of actions that are allowed in the parallel composition of probabilistic automata.

Finally we apply the renaming $[{}^{a}/_{(a,a')};\ {}^{a}/_{(a,\perp')};\ {}^{a}/_{(\perp,a')}]_{a\in Act}$, and obtain $((\widehat{\mathbb{S}} \parallel \widehat{\mathbb{T}}[{}^{a'}/_{a}]_{a\in Act\cup\perp}) \wedge \mathsf{S}_A)\,[{}^{a}/_{(a,a')};\ {}^{a}/_{(a,\perp')};\ {}^{a}/_{(\perp,a')}]_{a\in Act}$.

The bijection will be taken as the mapping that takes a state in $\widehat{\mathbb{S} \parallel \mathbb{T}}$ to the equivalent state in $((\widehat{\mathbb{S}} \parallel \widehat{\mathbb{T}}[{}^{a'}/_{a}]_{a\in Act\cup\perp}) \wedge \mathsf{S}_A)\,[{}^{a}/_{(a,a')};\ {}^{a}/_{(a,\perp')};\ {}^{a}/_{(\perp,a')}]_{a\in Act}$. The bijection f is defined, for states allowed by the parallel composition of PAs, as follows:

- $i \in \{1,\ldots,k_1k_2\}$ is mapped into $\{(1,1),\ldots,(k_1,k_2)\}$ by $i \mapsto (((i-1)\operatorname{div} k_2)+1, ((i-1) \bmod k_2)+1)$,
- $i \in \{k_1k_2+1,\ldots,2k_1k_2\}$ is mapped into $\{(k_1+1,k_2+1),\ldots,(2k_1,2k_2)\}$ by $i \mapsto (((i-1)\operatorname{div} k_2)+1, ((i-1) \bmod k_2)+1+k_2)$, and
- $i \in \{2k_1k_2+1,\ldots,2k_1k_2+l\}$ is mapped injectively into $\{k_1+1,\ldots,2k_1\}\times\{2k_2+1,\ldots,2k_2+l_2\}\cup\{2k_1+1,\ldots,2k_1+l_1\}\times\{k_2+1,\ldots,2k_2\}\cup\{2k_1+1,\ldots,2k_1+l_1\}\times\{2k_2+1,\ldots,2k_2+l_2\}$. This is done such that, $f(2k_1k_2+p)=(2k_1+q,2k_2+r)$, if both $2k_1k_2+p$ and $(2k_1+q,2k_2+r)$ are labeled with an $a\in A$ and there exists $s\in\{s_1,\ldots,s_{k_1}\}$ and $t\in\{t_1,\ldots,t_{k_2}\}$, such that $s\xrightarrow{a}{}^{S}\pi$ and $t\xrightarrow{a}{}^{T}\rho$ and $(s,t)\xrightarrow{a}\pi\otimes\rho$ and the constraint functions of $2k_1+q$ and $2k_2+r$ are satisfied by π and ρ, respectively and the constraint function of $2k_1k_2+p$ is satisfied by $\pi\otimes\rho$. Similarly, $f(2k_1k_2+p)=(k_1+q,2k_2+r)$, if both $2k_1k_2+p$ and $(k_1+q,2k_2+r)$ are labeled with an $a\notin A$ and there exists $s\in\{s_1,\ldots,s_{k_1}\}$ and $t\in\{t_1,\ldots,t_{k_2}\}$, such that $t\xrightarrow{a}{}^{T}\rho$ and $\pi=[{}^{s}_{\{s_1,\ldots,s_k\}}]$ and $(s,t)\xrightarrow{a}\pi\otimes\rho$.

From the above, is it clear that for type-3 states i, i and $f(i)$ will have equivalent constraint functions. For a type-1 state i, a distribution giving probability 1 to $i+k_1k_2$ is allowed. In $f(i)$ the same distribution is allowed, since, in the constraints of $\widehat{\mathbb{S}}$ and $\widehat{\mathbb{T}}$, distributions allowing $\mathbb{S}$ and $\mathbb{T}$ to idle, are allowed. Same argument holds for type-2 states.

It is then clear, that for a state v of $\widehat{\mathbb{S} \parallel \mathbb{T}}$, v and $f(v)$ have equivalent constraint functions. □

Interestingly, the precongruence property for parallel composition of PAs is obtained as a corollary of the above two reduction theorems and Theorem. 7.

Another, very similar, but slightly more complicated, encoding exists, for which weak refinement coincides with *probabilistic* simulation. Consider a PA $\mathbb{S}=(S,Act,\rightarrow,s_1)$, where $S=\{s_1,\ldots,s_k\}$. Let $\{(s^1,a_1),\ldots,(s^l,a_l)\}=\{(s,a)\mid s\in S\wedge a\in Act\}$. The corresponding CMC is

$$\check{\mathbb{S}}=(\{1,\ldots,2k+l\},1,\check{\varphi},Act\ \cup\perp,\check{V}\})\ ,$$

where $\perp$ is a fresh symbol not in Act . We have three types of states (see Figure 19). type-1 states, $\{1,\ldots,k\}$, correspond directly to states $\{s_1,\ldots,s_k\}$—their distribution constraints encode the non-deterministic choice of action. type-2 states, $\{k+1,\ldots,2k\}$, represent the ability of a state to be idle. We will use them in parallel composition. type-3 states, $\{2k+1,\ldots,2k+l\}$, encode choice of a probability distribution as a linear combination of distributions allowed by the automaton.

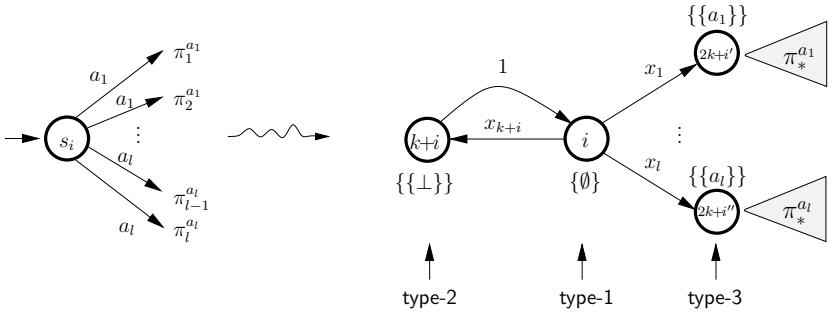


Figure 19: An attempt to visualize the second encoding. π_*^a denotes a constraint expressing a probability vector that is a linear combination of all probability distributions labeled by a. Below this is formalized as $\check{\varphi}(2k+i')(x)$.

The valuation functions are given by:

$$\begin{aligned}
&\check{V}(i) = \{\emptyset\} && \text{for } 1 \leq i \leq k \\
&\check{V}(k+i) = \{\{\bot\}\} && \text{for } 1 \leq i \leq k \\
&\check{V}(2k+i') = \{\{a_{i'}\}\} && \text{for } 1 \leq i' \leq l
\end{aligned}$$

and

$$\begin{aligned}
&\check{\varphi}(i)(x) \text{ is } x_{k+i} = 1 \text{ or } \exists 1 \leq i' \leq l.\ x_{2k+i'} = 1 \wedge s^{i'} = s_i && \text{for } 1 \leq i \leq k \text{ (type-1)} \\
&\check{\varphi}(k+i)(x) \text{ is } x_i = 1 && \text{for } 1 \leq i \leq k \text{ (type-2)} \\
&\check{\varphi}(2k+i')(x) \text{ is } \exists \lambda \in \mathrm{Dist}(1, \ldots, |\boldsymbol{\pi}|).\ x = \boldsymbol{\pi}\lambda && \text{for } 1 \leq i' \leq l \text{ (type-3)},
\end{aligned}$$

where $\boldsymbol{\pi} = \{\pi \mid s^j \xrightarrow{a_j} \pi\}$. Technically speaking $\boldsymbol{\pi}$ is a matrix, whose columns are distributions π. We write $|\boldsymbol{\pi}|$ for the number of columns in $\boldsymbol{\pi}$. Additionally x is implicitly required to be a probability distribution over $\{1, \ldots, 2k+l\}$.

Observe that $\widehat{\mathbb{S}}$ is only polynomially larger than $\mathbb{S}$.

Lemma 24 (Soundness)**.** *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\check{\mathbb{S}}$ weakly refines $\check{\mathbb{T}}$, we have that $\mathbb{T}$ probabilistically simulates $\mathbb{S}$.*

Proof. Let $\mathbb{S} = (S, Act, \rightarrow^S, s_1)$ and $\mathbb{T} = (T, Act, \rightarrow^T, t_1)$, with $S = \{s_1, \ldots, s_{k_1}\}$ and $T = \{t_1, \ldots, t_{k_2}\}$. In the proof we write $\check{\varphi}$ to refer to the constraint function of $\check{\mathbb{S}}$, and $\check{\rho}$ to refer to the constraint function of $\check{\mathbb{T}}$. Also l_1 and l_2 are used to refer to the number of combinations of state-actions of respectively $\check{\mathbb{S}}$ and $\check{\mathbb{T}}$. Finally q_i and r_j are used to range over states in S (respectively in T), when s_i and t_j are bound to some concrete value.

Let $\mathcal{R} \in \{1, \ldots, 2k_1 + l_1\} \times \{1, \ldots, 2k_2 + l_2\}$ be a weak refinement relation between $\check{\mathbb{S}}$ and $\check{\mathbb{T}}$, witnessing the assumption of the lemma. The proof proceeds by showing that

$$\mathcal{Q} = \{(s_i, t_j) \mid (i, j) \in R \wedge 1 \leq i \leq k_1 \wedge 1 \leq j \leq k_2\}$$

is a probabilistic simulation relation between $\mathbb{S}$ and $\mathbb{T}$.

We apply the usual coinductive proof technique. Take $(s_i, t_j) \in \mathcal{Q}$. Let $\pi \in \text{Dist}(S)$ be such that $s_i \xrightarrow{a} \pi$, and $(s^{i'}, a_{i'}) = (s_i, a)$.[1]

By construction of the encoding we know that any probability distribution x satisfying $\varphi(i)(x)$ is a point distribution, and x such that $x_{2k+i'} = 1$ is possible. So consider such a distribution x. Since $(i, j) \in \mathcal{R}$ we know that there exists a correspondence matrix $\Delta \in [0,1]^{(2k_1+l_1)\times(2k_2+l_2)}$ such that $\psi(j)(x\Delta)$ holds. Moreover $x\Delta$ must be a point distribution by construction of the encoding. So $(x\Delta)_{2k_2+j'} = 1$ for some $1 \leq j' \leq l_2$. And, by refinement again, we get that valuation functions for $2k_1 + i'$ and for $2k_2 + j'$ both return $\{\{a\}\}$ and that $(2k_1 + i', 2k_2 + j') \in R$.

But $\check{\mathbb{T}}$ is also constructed using the encoding, so it necessarily is that $t_j \xrightarrow{a} \rho$ for some $\rho \in \text{Dist}(T)$.

Observe that $\varphi(2k_1+i')(\pi)$ holds, because π is always a convex linear combination of a set of vectors containing it. Since $(2k_1+i', 2k_2+j') \in R$, there exists a correspondence matrix $\Delta' \in [0,1]^{(2k_1+l_1)\times(2k_2+l_2)}$ such that $\psi(2k_2 + j')(\pi\Delta')$ holds. The latter implies that $\pi\Delta'$ is a linear combination of vectors in $\boldsymbol{\rho} = \{\rho \mid t_j \xrightarrow{a} \rho\}$.

It remains to show that $\pi\mathcal{R}^*(\pi\Delta')$. Take $\alpha_{q_i,q_j} = \pi_i\Delta'_{ij}$. We first argue that $\alpha \in \text{Dist}(S \times T)$. Since each row of a correspondence matrix sums up to 1, we have that $\pi_i\Delta'_{ij} \in [0,1]$ for all i, j. Also $\sum_{i=1}^{k_1}\sum_{j=1}^{k^2}\pi_i\Delta'_{ij} = \sum_{i=1}^{k_1}\pi_i = 1$.

Consider $\alpha_{q_i,T} = \sum_{j=1}^{k_2}\alpha_{q_i,t_j} = \sum_{j=1}^{k_2}\pi_i\Delta'_{ij} = \pi_i\sum_{j=1}^{k_2}\Delta'_{ij} = \pi_i$ as required by $\pi\mathcal{R}^*(\pi\Delta')$.

Now consider $\alpha_{S,r_j} = \sum_{i=1}^{k_1}\alpha_{s_i,r_j} = \sum_{i=1}^{k_1}\pi_i\Delta'_{ij} = (\pi\Delta')_j$ as required by $\pi\mathcal{R}^*(\pi\Delta')$.

Now if $\alpha_{q_i,r_j} \neq 0$, then $\Delta'_{ij} \neq 0$, which in turn with refinement of $2k_2 + j'$ by $2k_1 + i'$ implies that $(i, j) \in \mathcal{R}$, and furthermore $(s_i, s_j) \in \mathcal{Q}$ by construction, as required by $\pi\mathcal{R}^*(\pi\Delta')$. This finishes the proof.

□

Lemma 25 (Completeness)**.** *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\mathbb{T}$ probabilistically simulates $\mathbb{S}$, we have that $\check{\mathbb{S}}$ weakly refines $\check{\mathbb{T}}$.*

Proof. Let $\mathbb{S} = (S, Act, \rightarrow^S, s_1)$ and $\mathbb{T} = (T, Act, \rightarrow^T, t_1)$, with $S = \{s_1, \ldots, s_{k_1}\}$ and $T = \{t_1, \ldots, t_{k_2}\}$. Let $\mathcal{Q} \subseteq S \times T$ be the probabilistic simulation relation between $\mathbb{S}$ and $\mathbb{T}$, witnessing the assumption of the lemma.

The proof proceeds by showing that a relation $\mathcal{R} \subseteq \{1, \ldots, 2k_1+l_1\}\times\{1, \ldots, 2k_2+l_2\}$ is a weak refinement relation between $\check{\mathbb{S}}$ and $\check{\mathbb{T}}$.

Take the following candidate for $\mathcal{R}$:

$$\mathcal{R}_1 = \{(i, j) \mid (s_i, t_j) \in \mathcal{Q}\}$$
$$\mathcal{R}_2 = \{(k_1 + i, k_2 + j) \mid (s_i, t_j) \in \mathcal{Q}\}$$
$$\mathcal{R}_3 = \{(2k_1 + i', 2k_2 + j') \mid (s_i, t_j) \in R \wedge s_i = s^{i'} \wedge t_j = t^{j'}\}$$
$$\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$$

[1] The equality binds i' to be the index of (s_i, a) on the list of state-action pairs in the encoding of $\mathbb{S}$.

We apply the usual coinductive proof technique.

Case 1. Take $(i,j) \in \mathcal{R}_1$ and x satisfying $\varphi(i)(x)$. We know that x can only be a point-distribution. If $x_{k_1+i} = 1$, then we take Δ such that $\Delta_{k_1+i,k_2+j} = 1$ (and Δ is zero for all other elements). Clearly Δ is a correspondence matrix. Moreover $x\Delta$ is a point distribution with 1 in the (k_2+j)th position, so $\psi(j)(x\Delta)$ holds by construction of the encoding (see first case in encoding of constraints). Also $(k_1+i, k_2+j) \in R_2$ since $(s_i, t_j) \in \mathcal{Q}$.

If $x_{2k_1+i'} = 1$, then it means that $s_i \xrightarrow{\check{V}(i)} \pi$ for some π and action $\check{V}(i)$. But then, since $(s_i, t_j) \in \mathcal{Q}$, it is possible that $t_j \xrightarrow{\check{V}(i)}_c \rho$,for some distribution ρ. Let j' be such that $t_j = t^{j'}$ and $a_{j'} = \check{V}(i)$. Take a correspondence matrix Δ such that $\Delta_{2k_1+i',2k_2+j'} = 1$ (and Δ is zero for all other elements). We have that $x\Delta$ is a point distribution with 1 in the $(2k_2+j')$th position, so $\psi(j)(x\Delta)$ holds by construction of encoding resulting in j (see first case in encoding of constraints). Also $(2k_1+i', 2k_2+j') \in R_3 \subseteq R$ by definition of $\mathcal{R}_3$.

Case 2. Take $(k_1+i, k_2+j) \in R_2$. The argument is almost identical to the first sub-case in Case 1. We omit it here.

Case 3. Take $(2k_1+i', 2k_2+j') \in R_3$ and x satisfying $\varphi(2k_1+i')(x)$. Let $s_i = s^{i'}$ and $t_j = t^{j'}$. By $\mathcal{R}_3$ we know that $(s_i, t_j) \in \mathcal{Q}$. By construction of the encoding $s_i \xrightarrow{\check{V}(2k_1+i')} x$ and furthermore $t_j \xrightarrow{\check{V}(2k_1+i')}_c \rho$, where $\rho = \boldsymbol{\rho}\lambda$ for some probability distribution $\lambda \in \mathrm{Dist}(1, \ldots, |\boldsymbol{\rho}|)$. Clearly $\psi(2k_2+j')(\rho) = 1$. It remains to check that π can be correspondence to ρ.

To this end consider a correspondence matrix Δ such that

$$\Delta_{ij} = \begin{cases} \alpha_{s_i,t_j}/x_i & \text{if } x_i \neq 0 \text{ and } i \leq k_1, j \leq k_2 \\ 0 & \text{otherwise} \end{cases}$$

Now $(x\Delta)_j = \sum_{i=1}^{2k_1+l_1} x_i\Delta_{ij} = \sum_{i=1}^{k_1} x_i\alpha_{s_i,t_j}/x_i = \sum_{i=1}^{k_1} \alpha_{s_i,t_j} = \alpha_{S,t_j} = \rho_j$ by $xR^*\rho$ (this discussion only holds for $j \leq k_2$, but the remaining elements are zero, which is easy to argue for. Also somewhat sloppily we ignored the possibility of division by zero – indeed it cannot happen since for $x_i = 0$ we said that Δ_{ij} is simply zero). Effectively $x\Delta = \rho$, so it satisfies $\psi(2k_2+j')$. Valuations obviously match.

Moreover if $\Delta_{ij} \neq 0$, then $\alpha_{s_i,t_j} \neq 0$ and $(s_i, t_j) \in \mathcal{Q}$ and then $(i,j) \in \mathcal{R}_1 \subseteq \mathcal{R}$, which finishes the proof. □

Theorem 26 is a corollary from the above two lemmas.

Theorem 26. $\mathbb{T}$ *probabilistically simulates* $\mathbb{S}$ *iff* $\check{\mathbb{S}}$ *weakly refines* $\check{\mathbb{T}}$.

Similarly, we obtain a precongruence with probabilistic simulation using a suitable encoding—a good example how CMCs can be used to study properties of simpler languages in a generic way.

12 Related Work and Concluding Remarks

We have presented CMCs—a new model for representing a possibly infinite family of MCs. Unlike the previous attempts [46, 48], our model is closed under many design

operations, including composition, conjunction, determinization and normalization. We have studied these operations as well as several classical compositional reasoning properties, showing that, among others, the CMC specification theory is equipped with a complete refinement relation (for deterministic specifications), which naturally interacts with parallel composition, synchronization and conjunction. We have also demonstrated how our framework can be used to obtain properties for less expressive languages, by using reductions. In particular, we have exemplified this for probabilistic automata with simulation and probabilistic simulation of Segala.

Two recent contributions [48, 49] are related to our work. Fecher et al. [48] propose a model checking procedure for PCTL [98] and Interval Markov Chains (other procedures recently appeared in [77, 78]), which is based on weak refinement. However, our objective is not to use CMCs within a model checking procedure for probabilistic systems, but rather to benefit from it as a specification theory.

Very recently Katoen and coauthors [49] have extended Fecher's work to *Interactive* Markov Chains, a model for performance evaluation [99, 100]. Their abstraction uses the continuous time version of IMCs [47] augmented with may and must transitions, very much in the spirit of [42]. Parallel composition is defined and studied for this abstraction, however conjunction has been studied neither in [48] nor in [49].

Over the years process algebraic frameworks have been proposed for describing and analyzing probabilistic systems based on Markov Chains (MCs) and Markov Decision Processes [29, 68, 92]. Also a variety of probabilistic logics have been developed for expressing properties of such systems, e.g., PCTL [18]. Both traditions support refinement between specifications using various notions of probabilistic simulation [46, 48] and, respectively, logical entailment [101]. Whereas the process algebraic approach favors parallel composition, the logical approach favors conjunction. Neither of the two supports *both* conjunction and parallel composition.

In mathematics the abstraction of Markov set-chains [69] lies very close to IMCs. It has been, for instance, used to approximate dynamics of hybrid systems [70]. The latter defines the intervals on the transition probabilities, while the former uses matrix intervals in the transition matrix space, which allows reasoning about the abstraction using linear algebra. Technically a Markov set-chain is an explicit enumeration of all the implementations of an IMC. While these works are clearly related to ours, we shall observe that like IMCs, these models are not closed under conjunction/composition.

In controller synthesis a notion of *Constrained Markov Decision Processes* (CMDPs) has been introduced. The similarity of name to CMCs is purely coincidental. In particular CMDPs are not a generalization/abstraction of CMCs. CMDPs, as described by Altman [102], are Markov Decision Processes annotated with several cost functions. They are used to synthesize probabilistic schedulers that optimize one cost function under a constraint over the other functions. Thus they are not a specification theory or an abstraction in the same sense as CMCs are.

As a future work, it would be of interest to design, implement and evaluate efficient algorithms for procedures outlined in this paper. We would also like to define a quotient relation for CMCs, presumably building on results presented in [103]. The quotienting

operation is of particular importance for component reuse [36, 41, 82, 104, 105]. One could also investigate the applicability of our approach in model checking procedures, in the same style as Fecher and coauthors have used IMCs for model checking PCTL [48]. Another promising direction would be to mix our results with those we recently obtained for timed specifications [45, 106, 107], hence leading to the first theory for specification of timed probabilistic systems [108]. We should also investigate more quantitative versions of the refinement operation like this was done for contracts in [109]. Finally, it would be interesting to extend our composition operation by considering products of dependent probability distributions in the spirit of [110].

Paper C – New Results for Constraint Markov Chains

Benoit Delahaye
INRIA/IRISA, Rennes

Kim G. Larsen
Aalborg University

Axel Legay
INRIA/IRISA, Rennes

Mikkel L. Pedersen
Aalborg University

Andrzej Wasowski
IT University, Copenhagen

1 Abstract

This paper studies compositional reasoning theories for stochastic systems. A specification theory combines notions of specification and implementation with satisfaction and refinement relations, and a set of operators that together support stepwise design. One of the first behavioural specification theories introduced for stochastic systems is the one of Interval Markov Chains (IMCs), which are Markov Chains whose probability distributions are replaced by a conjunction of intervals. In this paper, we show that IMCs are not closed under conjunction, which gives a formal proof of a conjecture made in several recent works.

In order to leverage this problem, we suggested to work with *Constraint Markov Chains* (CMCs) that is another specification theory where intervals are replaced with general constraints. Contrary to IMCs, one can show that CMCs enjoy the good closure properties of a specification theory. In addition, we propose aggressive abstraction procedures for CMCs. Such abstractions can be used either to combat the state-space explosion problem, or to simplify complex constraints. In particular, one can show that, under some assumptions, the behavior of any CMC can be abstracted by an IMC.

Finally, we propose an algorithm for counter-example generation, in case a refinement of two CMCs does not hold. We present a tool that implements our results. Implementing CMCs is a complex process and relies on recent advances made in deci-

sion procedures for theory of reals.

2 Introduction

Modern systems are build from multiple loosely-coupled components that interact with each other. These components are often designed independently but following a common agreement on what the interface of each component should be. An interface describes the coupling of components, ie. all the interaction between them. As a consequence, mathematical foundations that allow to reason at the level of interfaces in order to infer global properties of the system are an active area of research known as *compositional design* [1]. Within this area specification theories provide a modeling language that allows designing, evolving and advisedly reusing components with formal guarantees. In a logical interpretation, interfaces are specifications and systems/components that implement a specification are models/implementations. There is an agreement that a good theory should support the following requirements:

1. *Consistency and Satisfaction.* It should be decidable whether a specification admits at least one implementation, and whether a system implements a specification.

2. *Refinement. Refinement* of specification expresses inclusion of sets of implementations, and therefore allows to compare richness and precision of specifications.

3. *Structural composition.* A theory should provide a combination operator on specifications, reflecting the standard composition of systems by, *e.g.* parallel product.

4. *Logical composition/conjunction.* Different aspects of systems are often specified by different teams. The issue of dealing with multiple aspects of multiple viewpoints is thus essential. It should be possible to represent several specifications (viewpoints) for the same system, and to combine them in a logical/conjunctive fashion.

5. *Incremental Design.* A theory should allow incremental design (composing/conjoining specifications in any order) and independent implementability (composable specifications can always be refined separately) [91].

For functional analysis of discrete-time non-probabilistic systems, the theory of Modal Transition Systems (MTS) [42] provides a specification formalism supporting refinement as well as conjunction and parallel composition. It has been recently applied to construct interface theories [82, 83], which are extensions of classical interface automata proposed by de Alfaro et al. [84, 85, 86].

When it comes to modeling of real-time communication protocols, the functional model of MTSs is no longer sufficient. In [106] we have proposed a suitable compositional reasoning framework based on timed games, together with a user-friendly tool supporting modeling and analysis of designs. Very recently, we have been able to use the tool to model an actual wireless sensor system [111].

As soon as systems include randomized algorithms, probabilistic protocols, or interact with physical environment, probabilistic models are required to reason about them. This is exacerbated by requirements for fault tolerance, when systems need to be analyzed quantitatively for the amount of failure they can tolerate, or for the delays that may appear. As Henzinger and Sifakis [1] point out, introducing probabilities into design theories allows assessing dependability of IT systems in the same manner as commonly practiced in other engineering disciplines.

Generalizing the notion of MTSs to the non-functional analysis of probabilistic systems, the formalism of Interval Markov Chains (IMCs) was introduced [46]; with notions of satisfaction and refinement generalizing probabilistic bisimulation. Informally, IMCs extend Markov Chains by labeling transitions with *intervals* of allowed probabilities rather than concrete probability values. Implementations of IMCs are Markov Chains (MCs) whose probabilistic distributions match the constraints induced by the intervals. IMCs are known to be an efficient model on which refinement and composition can be performed with efficient algorithms of linear algebra. Unfortunately, as argued with the help of several examples (see e.g., [63]), the expressive power of IMCs seems to be inadequate to support both logical and structural composition.

In a recent work [63], we suggested to leverage the problem by enriching the model of IMCs by replacing intervals with general constraints. Our new model, which we call *Constraint Markov Chains* (CMCs) is a foundation for component-based design of probabilistic systems. CMCs are a further extension of IMCs allowing rich constraints on the next-state probabilities from any state. The model comes together with a behavioural semantic for both logical and structural composition. Whereas linear constraints suffice for closure under conjunction, polynomial constraints are necessary for closure under parallel composition. We also provided constructs for refinement, consistency checking, logical *and* structural composition of CMC specifications – all indispensable ingredients of a compositional design methodology.

In this paper, we propose new results for CMCs. Our contributions are summarized hereafter.

- First, we give the first formal proof that IMCs are indeed not closed under conjunction. This proof, which involves complex reasoning on the structure of the conjunction, establishes CMCs as the first complete behavioural semantic based stochastic specification theory supporting both logical and structural composition.

- Second, we consider abstraction techniques for CMCs. Our first abstraction combines states and may be used to combat state-space explosion. Our second abstraction may simplify complex constraints by abstracting the CMC with an IMC. Under some assumptions, one can show that such IMC is the minimal and unique abstraction. Both abstractions are compositional, but incomparable.

- Last but not least, we propose an implementation of our specification theory. Our new tool, which we called APAC, relies on the Z3 solver [112] to solve complex

constraints. In addition, the tool proposes a series of new features relying on new theoretical results. This includes the computation of a witness when refinement does not hold. While still being a prototype, the tool has already been evaluated on several complex CMCs.

We believe that investment in user-friendly tools is essential for successful adoption of theoretical results in engineering practice. We consider our prototype as a stepping stone towards a design environment, which will allow running case studies of similar complexity and realism as we have achieved for real time systems [111].

Structure of the paper Section 3 introduces IMCs as well as the proof that the formalism is not closed under conjunction. Section 4 introduces CMCs and summarizes existing results obtained in [63]. Section 5 presents a series of abstraction techniques as well as a running example. The implementation is presented in Section 6, while Section 7 reports experiments using this implementation. Finally, Section 8 concludes the paper and discusses related and future work.

3 Interval Markov Chains are not closed under Conjunction

We first introduce *Markov Chains* (MCs) which is a well-known mathematical model for purely stochastic systems.

Definition 1 (Markov Chain)**.** *$P = \langle \{1,\dots,n\}, o, M, A, V\rangle$ is a* Markov Chain *if $\{1,\dots,n\}$ is a set of states containing the initial state o, A is a set of atomic propositions, $V : \{1,\dots,n\} \to 2^A$ is a state valuation, and $M \in [0,1]^{n\times n}$ is a probability transition matrix: $\sum_{j=1}^{n} M_{ij} = 1$ for $i = 1,\dots,n$.*

Interval Markov Chains (IMCs) have been introduced in [46]. IMCs are a finite representation for a possibly infinite set of Markov Chains. Roughly speaking, IMCs generalize MCs in that, instead of specifying a concrete transition matrix, they only constrain probability values in the matrix to remain in some intervals.

Definition 2 (Interval Markov Chain)**.** *An* Interval Markov Chain *is a tuple $S = \langle \{1,\dots,k\}, o, \varphi, A, V\rangle$, where $\{1,\dots,k\}$ is a set of states containing the initial state o, A is a set of atomic propositions, $V : \{1,\dots,k\} \to 2^A$ is a state valuation and $\varphi : \{1,\dots,k\} \to [0,1]^k \to \{0,1\}$ is a* constraint function *defining probability intervals for transitions. A vector x satisfies the constraint of state j, written $\varphi(j)(x) = 1$, iff x is a distribution: $x \in [0,1]^k$ and $\sum_{i=1}^{k} x_i = 1$; and each of its coordinates falls inside the corresponding interval: $x_i \in \varphi(j)(i)$.*

Later, we often use the following notation for the constraint φ. For all states i:

$$\varphi(i)(x) \equiv \ (x_1 \in I_{i,1}) \wedge (x_2 \in I_{i,2}) \wedge \ldots \wedge (x_k \in I_{i,k}) \wedge (\sum_{j=1}^{k} x_j = 1) \qquad (1)$$

where $I_{i,j}$ is the interval corresponding to the transition between states i and j.

Given two sets A_1 and A_2 such that $A_1 \subseteq A_2$ and a subset $X \subseteq A_2$, the notation $X{\downarrow_{A_1}}$ denotes the restriction of X to A_1, i.e. $X \cap A_1$.

Definition 3 (Satisfaction Relation (IMCs)). *Let $P = \langle \{1,\dots,n\}, o_P, M, A_P, V_P\rangle$ be an MC and $S = \langle \{1,\dots,k\}, o_S, \varphi, A_S, V_S\rangle$ be an IMC with $A_S \subseteq A_P$. Then $\mathcal{R} \subseteq \{1,\dots,n\} \times \{1,\dots,k\}$ is a* satisfaction relation *between states of P and S iff whenever $p\,\mathcal{R}\,u$ then*

1. *$V_P(p){\downarrow_{A_S}} = V_S(u)$, and*
2. *There exists a correspondence matrix $\Delta \in [0,1]^{n\times k}$ such that*
   - *for all $1 \le p' \le n$ with $M_{pp'} \neq 0$, $\sum_{j=1}^{k} \Delta_{p'j} = 1$;*
   - *$\varphi(u)(M_p\Delta)$ holds, and if $\Delta_{p'u'} > 0$ then $p'\,\mathcal{R}\,u'$.*

We write $P \models S$ iff there exists a satisfaction relation relating o_P and o_S, and call P an *implementation* of S. The set of all implementations of S is given by $[\![S]\!] \equiv \{P \mid P \models S\}$.

The weak refinement relation syntactically relates IMCs S_1 and S_2 if (roughly) any implementation satisfying S_1 also satisfies S_2:

Definition 4 (Weak Refinement). *Let $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be IMCs with $A_2 \subseteq A_1$. The relation $\mathcal{R} \subseteq \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ is a* weak refinement relation *iff $v\,\mathcal{R}\,u$ implies:*

1. *$V_1(v){\downarrow_{A_2}} = V_2(u)$ and*
2. *For any distribution $x \in [0,1]^{k_1}$ satisfying $\varphi_1(v)(x)$, there exists a correspondence matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that*
   - *for all S_1 states $1 \le i \le k_1$ if $x_i \neq 0$ then $\sum_{j=1}^{k_2} \Delta_{ij} = 1$ and*
   - *$\varphi_2(u)(x\Delta)$ holds and*
   - *if $\Delta_{v'u'} > 0$ then also $v'\,\mathcal{R}\,u'$.*

IMC S_1 (weakly) refines S_2, written $S_1 \preceq S_2$, iff $o_1\,\mathcal{R}\,o_2$.

So far, IMCs have been used as an abstraction formalism in various stochastic model checking algorithms [47]. We now show that IMCs are not closed under conjunction. This means that this formalism cannot be used as a specification theory. We start by introducing an example and then give a formal proof.

Example 1. *Consider the IMCs of Figure 1. S_1 specifies a behaviour of a user of a coffee machine. It prescribes that a typical user orders coffee with milk with probability within $[0, 0.5]$ and orders black coffee with probability in $[0.2, 0.7]$. Customers also buy tea with probability in the interval $[0, 0.5]$. Now the vendor of the machine delivers another specification, S_2, which prescribes that the machine is functioning only if coffee*

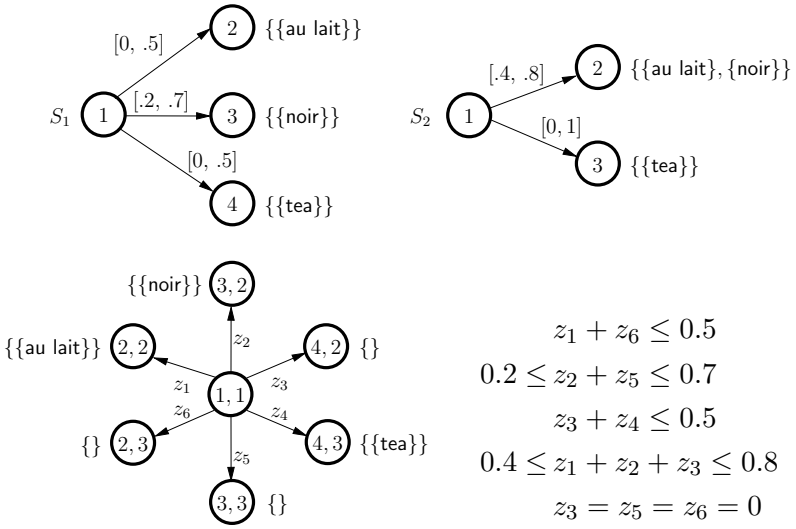


Figure 1: IMCs showing non-closure under conjunction. Top: the two specifications of different aspects of a coffee service. Bottom a conjunction expressed as a Markov Chain with linear constraints over probability values.

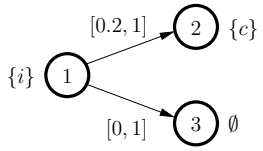
(white or black) is ordered with probability between 0.4 *and* 0.8*. Otherwise, the machine runs out of coffee powder too frequently, or the powder becomes too old. A conjunction of these two models would describe users who have use patterns compatible with this particular machine. In the bottom part of Figure 1 we present the structure of such a conjunction. States* $(2,3)$*,* $(3,3)$*, and* $(4,2)$ *are inconsistent and thus the corresponding probabilities must be zero:* $z_3 = z_5 = z_6 = 0$*. Now, attempting to express the conjunction* $S_1 \wedge S_2$ *as an IMC by a simple intersection of bounds gives* $0.4 \leq z_1 \leq 0.5$*,* $0.4 \leq z_2 \leq 0.7$*, and* $z_4 \leq 0.5$*. However, this naive construction is too coarse: whereas* $(z_1, z_2, z_3, z_4, z_5, z_6) = (0.5, 0.5, 0, 0, 0, 0))$ *satisfies the constraints the resulting overall probability of reaching a state satisfying* $\{\{\mathsf{au\ lait}\}, \{\mathsf{noir}\}\}$*, i.e.* $z_1+z_2+z_3 = 1$*, violates the upper bound of* 0.8 *specified in* S_2*.*

We now propose the main result of this section:

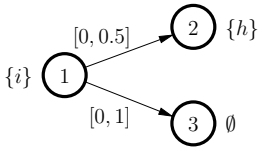
Theorem 1. *IMCs are not closed under conjunction.*

Proof. Consider IMCs S_1 and S_2 given in Figure 2a and 2b, respectively. These IMCs represent two requirements on a coffee machine. IMC $S_1 = \langle \{1,2,3\}, 1, \{i, c\}, \varphi_1, V_1 \rangle$ specifies that a good coffee machine should serve coffee at least 20% of requests ("coffee" represented by the atomic proposition c). IMC $S_2 = \langle \{1,2,3\}, 1, \{i, h\}, \varphi_2, V_2 \rangle$ specifies that a good coffee machine must serve hot drinks at most 50% of requests ("hot" represented by the atomic proposition h).

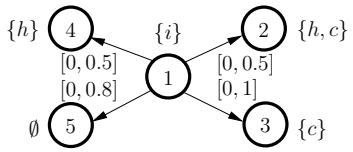
We claim that there exists no IMC accepting the intersection of the set of models of S_1 and S_2. The proof is by contradiction. Suppose that such an IMC $S = \langle Q, q_0, \{i, c, h\}, \varphi, V \rangle$ exists. We first observe that only the intial states in S_1 and S_2



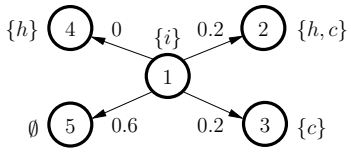
(a) IMC S_1 specifying a first requirement for the coffee machine



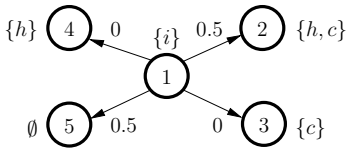
(b) IMC S_2 specifying a second requirement for the coffee machine



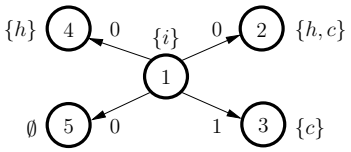
(c) IMC S_3 specifying a combination of the two requirements



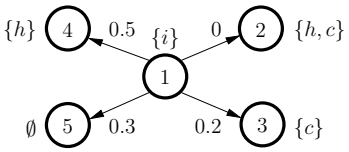
(d) MC I_1: A candidate implementation



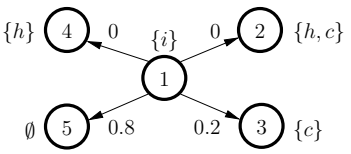
(e) MC I_2: A candidate implementation



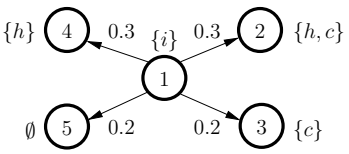
(f) MC I_3: A candidate implementation



(g) MC I_4: A candidate implementation



(h) MC I_5: A candidate implementation



(i) MC I_6: A candidate implementation

Figure 2: A counter-example illustrating non-closure of IMCs under conjunction.

have the atomic proposition i in their valuation, it is safe to suppose that the only accessible valuations in S are the empty set, $\{i\}, \{c\}, \{h\}$ and $\{h, c\}$. We partition the

set of states of S according to their valuations:

$$\begin{aligned} Q^i &= \{q \in Q \mid V(q) = \{i\}\} = \{q_0\} \\ Q^h &= \{q \in Q \mid V(q) = \{h\}\} = \{q^h_1, \ldots, q^h_{n_h}\} \\ Q^c &= \{q \in Q \mid V(q) = \{c\}\} = \{q^c_1, \ldots, q^c_{n_c}\} \\ Q^{hc} &= \{q \in Q \mid V(q) = \{h,c\}\} = \{q^{hc}_1, \ldots, q^{hc}_{n_{hc}}\} \\ Q^{\emptyset} &= \{q \in Q \mid V(q) = \emptyset\} = \{q^{\emptyset}_1, \ldots, q^{\emptyset}_{n_\emptyset}\} \end{aligned}$$

For all i and $X \in \{i, c, h, hc, \emptyset\}$, let $\varphi(q_0)(q^X_i)$ be the interval $[m^X_i, U^X_i]$, where both endpoints are variables constrained as below. By construction, we know that the implementation I_1, given in Figure 2d, is an implementation of both S_1 and S_2. As a consequence, it is also an implementation of S. We can thus deduce the following inequalities:

$$\sum_{i=1}^{n_{hc}} m^{hc}_i \leq 0.2 \qquad \sum_{i=1}^{n_c} m^c_i \leq 0.2 \qquad \sum_{i=1}^{n_h} m^h_i \leq 0 \qquad \sum_{i=1}^{n_\emptyset} m^\emptyset_i \leq 0.6 \tag{2}$$

Similarly, observe that MCs I_2, I_3, I_4 and I_5 of Figures 2e, 2f, 2g, and 2h are also implementations of both S_1 and S_2, which gives the following inequalities:

$$\sum_{i=1}^{n_{hc}} m^{hc}_i \leq 0, \qquad \sum_{i=1}^{n_{hc}} U^{hc}_i \geq 0.5 \qquad \sum_{i=1}^{n_c} m^c_i \leq 0, \qquad \sum_{i=1}^{n_c} U^c_i \geq 1 \tag{3}$$

$$\sum_{i=1}^{n_h} m^h_i \leq 0, \qquad \sum_{i=1}^{n_h} U^h_i \geq 0.5 \qquad \sum_{i=1}^{n_\emptyset} m^\emptyset_i \leq 0, \qquad \sum_{i=1}^{n_\emptyset} U^\emptyset_i \geq 0.8 \tag{4}$$

We will now show that there exists a model of S that does not satisfy S_1 or S_2, which will lead to a contradition, as we have assumed that S expresses the greatest lower bound of S_1 and S_2. For doing so, we will instantiate S with concrete distributions through refinement. Consider IMC $S_3 = \langle Q_3 = \{1,2,3,4,5\}, 1, \{i,c,h\}, \varphi_3, V_3\rangle$ given in Figure 2c. We show that S_3 weakly refines S. For this, we consider the relation $\mathcal{R} \subseteq Q_3 \times Q$ such that $1\,\mathcal{R}\,q_0$, $2\,\mathcal{R}\,q$ iff $q \in Q^{hc}$, $3\,\mathcal{R}\,q$ iff $q \in Q^c$, $4\,\mathcal{R}\,q$ iff $q \in Q^h$ and $5\,\mathcal{R}\,q$ iff $q \in Q^\emptyset$. We show that $\mathcal{R}$ is indeed a weak refinement relation by providing the correspondence matrices required to witness $1\,\mathcal{R}\,q_0$. The matrices for the other states can be obtained in a similar manner. Let $x = (x_1, x_2, x_3, x_4, x_5)$ be a transition vector such that $\varphi_3(1)(x)$ holds. Define Δ as follows: for all $1 \leq i \leq 5$, if $x_i = 0$, then for all

$q \in Q$ define $\Delta_{i,q} = 0$. Otherwise, define:

$$\Delta_{2,q} = 0 \text{ for all } q \notin Q^{hc} \qquad \Delta_{3,q} = 0 \text{ for all } q \notin Q^{c}$$
$$\Delta_{4,q} = 0 \text{ for all } q \notin Q^{h} \qquad \Delta_{5,q} = 0 \text{ for all } q \notin Q^{\emptyset}$$
$$\Delta_{2,q_i^{hc}} = \frac{1}{x_2}\left(m_i^{hc} + \frac{(U_i^{hc} - m_i^{hc})(x_2 - \sum_{j=1}^{n_{hc}} m_j^{hc})}{\sum_{j=1}^{n_{hc}}(U_j^{hc} - m_j^{hc})}\right)$$
$$\Delta_{3,q_i^{c}} = \frac{1}{x_3}\left(m_i^{c} + \frac{(U_i^{c} - m_i^{c})(x_3 - \sum_{j=1}^{n_{c}} m_j^{c})}{\sum_{j=1}^{n_{c}}(U_j^{c} - m_j^{c})}\right)$$
$$\Delta_{4,q_i^{h}} = \frac{1}{x_4}\left(m_i^{h} + \frac{(U_i^{h} - m_i^{h})(x_4 - \sum_{j=1}^{n_{h}} m_j^{h})}{\sum_{j=1}^{n_{h}}(U_j^{h} - m_j^{h})}\right)$$
$$\Delta_{5,q_i^{\emptyset}} = \frac{1}{x_5}\left(m_i^{\emptyset} + \frac{(U_i^{\emptyset} - m_i^{\emptyset})(x_5 - \sum_{j=1}^{n_{\emptyset}} m_j^{\emptyset})}{\sum_{j=1}^{n_{\emptyset}}(U_j^{\emptyset} - m_j^{\emptyset})}\right)$$

By construction, we have that for all $1 \leq i \leq 5$, if $x_i > 0$, then $\sum_{q \in Q} \Delta_{i,q} = 1$. Moreover, by 4 and since $\varphi_3(1)(x)$ holds, we also have that $\varphi(q_0)(x\Delta)$ holds. Thus Δ is a correspondence matrix and we have that $S_3 \preceq S$.

Since weak refinement for IMCs implies inclusion of sets of implementations [46], we have that all implementations of S_3 are also implementations of S, and consequently implementations of both S_1 and S_2. Consider now the implementation I_6 given in Figure 2i. It is obvious that I_6 satisfies S_3. However, the probability to reach a state with valuation $\{h\}$ or $\{hc\}$ in I_6 is 0.6, which means that I_6 does not satisfy S_2. As a consequence, S is not a greatest lower bound for S_1 and S_2. This concludes the proof. □

According to the above theorem, working with intervals is not enough to capture conjunction. A similar proof can be used to show that that intervals cannot capture structural composition, either. In the next section, we present Constraint Markov Chains, a new specification theory for stochastic systems where intervals are replaced by general constraints.

4 Constraint Markov Chains

We now introduce the concept of Constraint Markov Chains that was proposed in [63]. Unlike to IMCs, CMCs have all the closure properties expected of a specification theory.

Let A, B be sets of propositions with $A \subseteq B$. If $T \subseteq 2^B$, then $T{\downarrow_A} \equiv \{W{\downarrow_A} \mid W \in T\}$. For $W \subseteq A$ define the *extension of* W *to* B as $W{\uparrow^B} \equiv \{V \subseteq B \mid V{\downarrow_A} = W\}$, so the set of sets whose restriction to A is W. Lift it to sets of sets as follows: if $T \subseteq 2^A$ then $T{\uparrow^B} \equiv \{W \subseteq B \mid W{\downarrow_A} \in T\}$. Let $M, \Delta \in [0,1]^{n \times k}$ be two matrices and $x \in [0,1]^k$ be a vector. We write M_{ij} for the cell in ith row and jth column of M, M_p for the pth row of M, and x_i for the ith element of x. Finally, Δ is a *correspondence matrix* iff $0 \leq \sum_{j=1}^{k} \Delta_{ij} \leq 1$ for all $1 \leq i \leq n$.

Constraint Markov Chains (CMCs for short) are a finite representation for a possibly infinite set of MCs. Roughly speaking, CMCs generalize MCs in that, instead of specifying a concrete transition matrix, they only constrain probability values in the matrix. Constraints are modeled using a *characteristic function*, which for a given source state and a distribution of probabilities of leaving the state evaluates to 1 iff the distribution is permitted by the specification. Similarly, instead of a concrete valuation function for each state, a *constraint on valuations* is used. Here, a valuation is permitted iff it is contained in the set of admissible valuations of the specification.

Definition 5 (Constraint Markov Chain)**.** *A* Constraint Markov Chain *is a tuple* $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$*, where* $\{1, \ldots, k\}$ *is a set of states containing the initial state* o*,* A *is a set of atomic propositions,* $V : \{1, \ldots, k\} \to 2^{2^A}$ *is a set of admissible state valuations and* $\varphi : \{1, \ldots, k\} \to [0,1]^k \to \{0,1\}$ *is a* constraint function *such that if* $\varphi(j)(x) = 1$ *then the* x *vector is a probability distribution:* $x \in [0,1]^k$ *and* $\sum_{i=1}^{k} x_i = 1$*.*

An *Interval Markov Chain* is in fact a CMC whose constraint functions are represented by intervals, so for all $1 \leq i \leq k$ there exist constants α_i, β_i such that $\varphi(j)(x) = 1$ iff $\forall 1 \leq i \leq k,\ x_i \in [\alpha_i, \beta_i]$.

The notion of satisfaction, an extension of satisfaction for IMCs of Section 3, links Markov Chains and Constraint Markov Chains:

Definition 6 (Satisfaction Relation)**.** *Let* $P = \langle \{1, \ldots, n\}, o_P, M, A_P, V_P \rangle$ *be a MC and* $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ *be a CMC with* $A_S \subseteq A_P$*. Then* $\mathcal{R} \subseteq \{1, \ldots, n\} \times \{1, \ldots, k\}$ *is a* satisfaction relation *between states of* P *and* S *iff whenever* $p \,\mathcal{R}\, u$ *then*

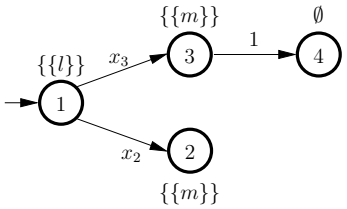
1. $V_P(p)\downarrow_{A_S} \in V_S(u)$*, and*
2. *There exists a correspondence matrix* $\Delta \in [0,1]^{n \times k}$ *such that*
   - *for all* $1 \leq p' \leq n$ *with* $M_{pp'} \neq 0$*,* $\sum_{j=1}^{k} \Delta_{p'j} = 1$*;*
   - $\varphi(u)(M_p \Delta)$ *holds and*
   - *if* $\Delta_{p'u'} \neq 0$ *then* $p' \,\mathcal{R}\, u'$*.*

CMC P *satisfies* S*, written* $P \models S$*, iff* $o_P \,\mathcal{R}\, o_S$*. The set of all implementations of* S *is denoted by* $[\![S]\!] = \{P \mid P \models S\}$*.*

Consistency A CMC S is *consistent* if it admits at least one implementation. We say that a state is inconsistent iff its set of admissible valuations is empty, or if its constraint is unsatisfiable. Consistency of all states implies consistency of the CMC, but a CMC having some inconsistent states may still be consistent.

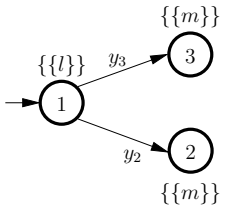
It is known [62] that consistency of a CMC can be decided with a *pruning algorithm*. The pruning operator β is defined as follows. We begin with the set of inconsistent states, and propagate inconsistency backwards using a co-inductive fixed-point algorithm.

Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC.



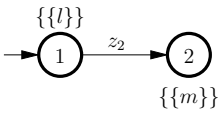
$\varphi(1)(x) \equiv (x_2 = 1 \vee x_3 = 1)$

(a) CMC S



$\varphi'(1)(y) \equiv \exists(x_1, x_2, x_3, x_4) : (x_4 = 0) \wedge (y_1 = x_1) \wedge (y_2 = x_2) \wedge (y_3 = x_3) \wedge \varphi(1)(x_1, x_2, x_3, x_4)$

(b) The first application of pruning: $\beta(S)$



$\varphi''(1)(z) \equiv \exists(y_1, y_2, y_3) : (y_3 = 0) \wedge (z_1 = y_1) \wedge (z_2 = y_2) \wedge \varphi'(1)(y_1, y_2, y_3)$

(c) The fixpoint of the pruning algorithm: $\beta^*(S)$

Figure 3: Two different steps of the pruning algorithm.

- If the initial state o is locally inconsistent, then let $\beta(S) = \emptyset$.
- If S does not contain locally inconsistent states, then $\beta(S) = S$.
- Otherwise, proceed in two steps. Let $k' < k$ be the number of locally consistent states. Then define a function $\nu : \{1, \ldots, k\} \to \{\bot, 1, \ldots, k'\}$. All inconsistent states are mapped to $\bot$, i.e. for all $1 \leq i \leq k$ take $\nu(i) = \bot$ iff $[(V(i) = \emptyset) \vee (\forall x \in [0,1]^k,\ \varphi(i)(x) = 0)]$. All remaining states are mapped injectively into $\{1, \ldots, k'\}$: $\nu(i) \neq \bot \implies \forall j \neq i,\ \nu(j) \neq \nu(i)$. Then let $\beta(S) = \langle \{1, \ldots, k'\}, \nu(o), \varphi', A, V' \}$, where $V'(i) = V(\nu^{-1}(i))$ and for all $1 \leq j \leq k'$ the constraint $\varphi'(j)(y_1, \ldots, y_{k'})$ is: $\exists x_1, \ldots, x_k$ such that

$$\big[\nu(q) = \bot \Rightarrow x_q = 0\big] \ \wedge \ \big[\forall 1 \leq l \leq k' : \ y_l = x_{\nu^{-1}(l)}\big] \ \wedge \ \big[\varphi(\nu^{-1}(j))(x_1, \ldots, x_k)\big]$$

The fixpoint of β preserves the set of implementations [62]: $[\![S]\!] = [\![\beta^*(S)]\!]$.

Example 2. *Figure 3 illustrates how pruning is performed for the CMC S in three steps.*

Single valuation normal form A CMC $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ is said to be in single valuation normal form, if all state valuations are singletons, ie. if $\forall i \in \{1, \ldots, k\}, |V(i) \models 1$. The single valuation normal form plays a central role in both determinism and abstraction. As is mentioned in [62], a consistent CMC, for which the

initial state o satisfies that $|V(o) \models 1$, can be transformed into a CMC in single valuation normal form with the same implementation set. The process, called *normalization*, is performed by the following algorithm:

Definition 7 (Normalization)**.** *Let $S = \langle\{1,\ldots,k\}, o, \varphi, A, V\rangle$ be a CMC. The normalization of S is only defined if o is in single valuation normal form (i.e. $|V(o) \models 1$) and if there exists a function $\mathcal{N} : \{1,\ldots,k\} \to 2^{\{1,\ldots,m\}}$ such that:*

1. $\{1,\ldots,m\} = \cup_{i\in\{1,\ldots,k\}}\mathcal{N}(i)$;
2. *For all* $1 \leq i \neq j \leq k$, $\mathcal{N}(i) \cap \mathcal{N}(j) = \emptyset$;
3. $\forall 1 \leq i \leq k$, $|\mathcal{N}(i)| = |V(i)|$;

Under these assumptions, the normalization of S is the CMC $\mathcal{N}(S) = \langle\{1,\ldots,m\}, o', \varphi', A, V'\rangle$ such that $\mathcal{N}(o) = o'$ and

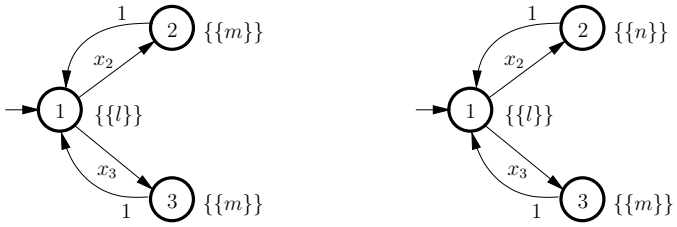
1. $\forall 1 \leq j \leq m$, $|V'(j)| = 1$;
2. $\forall 1 \leq i \leq k$, $V(i) = \cup_{u\in\mathcal{N}(i)}V'(u)$;
3. $\forall 1 \leq i \leq k$, $\forall u, v \in \mathcal{N}(i)$, $u \neq v \iff V'(u) \neq V'(v)$;
4. $\forall 1 \leq j \leq m.\, \varphi'(j)(x_1,\ldots x_m) = \varphi(\mathcal{N}^{-1}(j))(\sum_{u\in\mathcal{N}(1)} x_u, \ldots, \sum_{u\in\mathcal{N}(k)} x_u)$.

Comparing specifications is central to stepwise design methodologies. Usually specifications are compared using a *refinement* relation. Roughly, if S_1 refines S_2, then any model of S_1 is also a model of S_2. We recall two syntactic notions of refinement for CMCs [63] that extend the refinements for IMCs [46, 48] presented above. We begin with the *strong refinement*:

Definition 8 (Strong Refinement)**.** *Let $S_1 = \langle\{1,\ldots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle\{1,\ldots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be CMCs with $A_2 \subseteq A_1$. A relation $\mathcal{R} \subseteq \{1,\ldots,k_1\} \times \{1,\ldots,k_2\}$ is a* strong refinement relation *between states of S_1 and S_2 iff whenever $v\,\mathcal{R}\,u$ then*

1. $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$, *and*
2. *There exists a correspondence matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that for all probability distribution vectors $x \in [0,1]^{k_1}$ if $\varphi_1(v)(x)$ holds then*
   - *for all S_1 states $1 \leq i \leq k_1$, $x_i \neq 0 \implies \sum_{j=1}^{k_2}\Delta_{ij} = 1$;*
   - *$\varphi_2(u)(x\Delta)$ holds and*
   - *if $\Delta_{v'u'} \neq 0$ then $v'\,\mathcal{R}\,u'$.*

We say that S_1 strongly refines S_2 iff $o_1\,\mathcal{R}\,o_2$.



$\varphi_1(1)(x) \equiv x_2 + x_3 = 1$ $\varphi_2(1)(x) \equiv x_2 + x_3 = 1$

(a) CMC S_1 is non-deterministic **(b)** CMC S_2 is deterministic

Figure 4: A non-deterministic CMC and a deterministic CMC

Strong refinement imposes a "fixed-in-advance" correspondence matrix regardless of the probability distribution satisfying the constraint function. In contrast, the *weak refinement* allows choosing a different correspondence matrix for each probability distribution satisfying the constraint:

Definition 9 (Weak Refinement)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_2 \subseteq A_1$*. The relation* $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ *is a* weak refinement relation *iff* $v \,\mathcal{R}\, u$ *implies:*

1. $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$ *and*
2. *For any distribution* $x \in [0,1]^{k_1}$ *satisfying* $\varphi_1(v)(x)$*, there exists a correspondence matrix* $\Delta \in [0,1]^{k_1 \times k_2}$ *such that*
   - *for all* S_1 *states* $1 \leq i \leq k_1$*,* $x_i \neq 0 \implies \sum_{j=1}^{k_2} \Delta_{ij} = 1$*;*
   - $\varphi_2(u)(x\Delta)$ *holds and*
   - $\Delta_{v'u'} \neq 0 \Rightarrow v' \,\mathcal{R}\, u'$*.*

CMC S_1 *(weakly) refines* S_2*, written* $S_1 \preceq S_2$*, iff* $o_1 \,\mathcal{R}\, o_2$*.*

In [63], we have shown that strong refinement implies weak refinement that also implies implementation set inclusion. We also showed that the reverse of those implications does not hold. The exception is for deterministic CMCs, which are introduced hereafter.

Determinism A CMC S is *deterministic* iff for every state i, states reachable from i have pairwise disjoint admissible valuations. CMCs are not closed under determinization. More precisely, there exists a non-deterministic CMC for which there is no deterministic CMC accepting the same set of models. In [63], we have proposed a determinization algorithm for CMCs, which computes a deterministic CMC accepting all models of the original CMC.

Example 3. *Figure 4 shows a non-deterministic and a deterministic CMC. Indeed the only discrepancy of* S_2 *with respect to* S_1 *is that* $V_2(2) = \{\{n\}\}$*, and this ensures that the states reachable from state* 1 *of* S_2 *has pairwise disjoint valuations.*

Theorem 2 ([63]). *For deterministic CMCs, strong refinement coincides with weak refinement and inclusion set implementation.*

A good compositional theory comes together with two composition operations. The first composition is structural and allows to combine components. The second operation, which is often called conjunction, is logical and allows to take the intersection of a set of requirements.

Structural composition This composition mimics the classical composition on transition systems at the specification level. We first present the composition between two MCs and then the one between CMCs.

Definition 10 (Parallel Composition of MCs). *Let* $P_1 = \langle \{1, \dots, n_1\}, o_1, M', A_1, V_1 \rangle$ *and* $P_2 = \langle \{1, \dots, n_2\}, o_2, M'', A_2, V_2 \rangle$ *be two MCs with* $A_1 \cap A_2 = \emptyset$. *The parallel composition of* P_1 *and* P_2 *is the MC* $P_1 \parallel P_2 = \langle \{1, \dots, n_1\} \times \{1, \dots, n_2\}, (o_1, o_2), M, A_1 \cup A_2, V \rangle$ *where:* $M \in [0,1]^{(n_1 \times n_2) \times (n_1 \times n_2)}$ *is such that* $M_{(p,q)(r,s)} = M'_{pr} M''_{qs}$; *and* $V((p,q)) = V_1(p) \cup V_2(q)$.

Definition 11 (Parallel Composition of CMCs). *Let* $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_1 \cap A_2 = \emptyset$. *The parallel composition of* S_1 *and* S_2 *is the CMC* $S_1 \parallel S_2 = \langle \{1, \dots, k_1\} \times \{1, \dots, k_2\}, (o_1, o_2), \varphi, A_1 \cup A_2, V \rangle$, *where* $\varphi((u,v))(z_{1,1}, z_{1,2}, \dots z_{2,1}, \dots, z_{k_1,k_2}) = \exists x_1, \dots, x_{k_1}, y_1, \dots, y_{k_2} \in [0,1]$ *such that* $\forall (i,j) \in \{1, \dots, k_1\} \times \{1, \dots, k_2\}$ *we have* $z_{i,j} = x_i y_j$ *and* $\varphi_1(u)(x_1, \dots, x_{k_1}) = \varphi_2(v)(y_1, \dots, y_{k_2}) = 1$; *Finally,* $V((u,v)) = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$.

By inspecting the above definition, the reader shall intuitively understand that the composition of two IMCs[1] is generally not an IMC. Examples are presented in [63], and a formal proof can be obtained by following the proof for conjunction we introduced in Section 3.

It is known [63], that structural composition has the property of independent implementability:

Theorem 3. *If* S'_1, S'_2, S_1, S_2 *are CMCs then* $S'_1 \preceq S_1$ *and* $S'_2 \preceq S_2$ *implies* $S'_1 \parallel S'_2 \preceq S_1 \parallel S_2$, *so the weak refinement is a precongruence with respect to parallel composition. Consequently, for any MCs* P_1 *and* P_2 *we have that* $P_1 \models S_1 \wedge P_2 \models S_2$ *implies* $P_1 \parallel P_2 \models S_1 \parallel S_2$.

Observe that one cannot combine CMCs sharing atomic propositions. Indeed, this would create a dependency between the probability distributions. In order to synchronize an atomic action, one will have to combine structural composition with the logical composition described hereafter. This reflects the principle of separation of concerns for composition, introduced for transition systems in [113]. It is also, at large, followed by formalisms like Interface Automata [84], which apply inconsistency elimination after computing the composition.

[1] As IMCs are subsets of CMCs, the composition of two IMCs is defined as their CMC composition.

Logical composition This operation, also called conjunction, combines requirements of several specifications.

Definition 12 (Conjunction)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be two CMCs. The conjunction of* S_1 *and* S_2*, written* $S_1 \wedge S_2$*, is the CMC* $S = \langle \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}, (o_1, o_2), \varphi, A, V \rangle$ *with* $A = A_1 \cup A_2$*,* $V((u, v)) = V_1(u)\uparrow^A \cap V_2(v)\uparrow^A$*, and*

$$\varphi((u,v))(x_{1,1}, x_{1,2}, \ldots, x_{2,1}, \ldots, x_{k_1,k_2}) \equiv \begin{array}{l} \varphi_1(u)(\sum_{j=1}^{k_2} x_{1,j}, \ldots, \sum_{j=1}^{k_2} x_{k_1,j}) \wedge \\ \varphi_2(v)(\sum_{i=1}^{k_1} x_{i,1}, \ldots, \sum_{i=1}^{k_1} x_{i,k_2}). \end{array}$$

Conjunction may introduce inconsistent states. Indeed, the intersection between the sets of valuations of two states may be empty. Conjunction should thus normally be followed by applying the pruning algorithm. In [63], we proved that conjunction of two specifications coincides with their greatest lower bound with respect to the weak refinement (also called *shared refinement*).

Theorem 4. *Let* S_1*,* S_2 *and* S_3 *be three CMCs. We have (a)* $((S_1 \wedge S_2) \preceq S_1)$ *and* $((S_1 \wedge S_2) \preceq S_2)$ *and (b) if* $(S_3 \preceq S_1)$ *and* $(S_3 \preceq S_2)$*, then* $S_3 \preceq (S_1 \wedge S_2)$*.*

The first consequence of the above theorem is that conjunction with another specification is a monotonic operator with respect to weak refinement. Furthermore, the set of implementations of a conjunction of two specifications S_1 and S_2 coincides with the intersection of implementation sets of S_1 and S_2 (the greatest lower bound in the lattice of implementation sets).

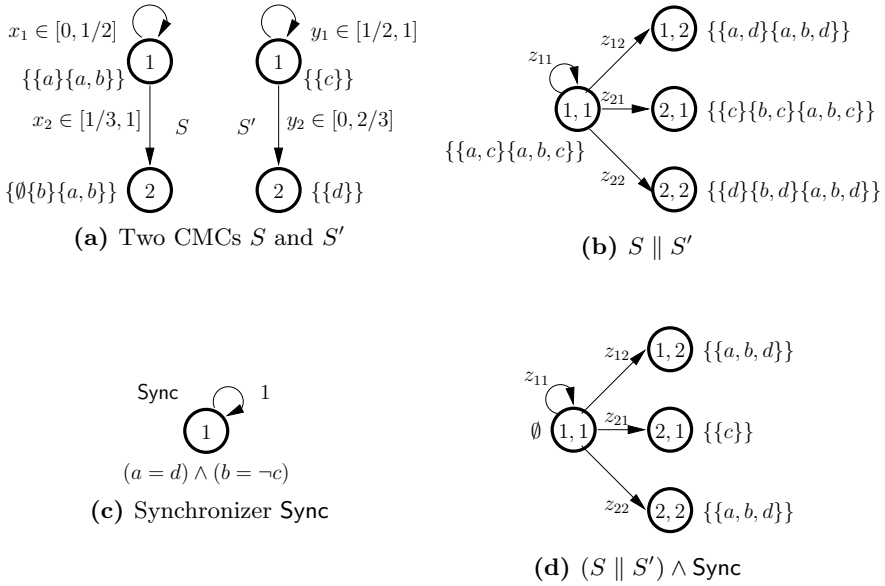
Combining compositions One shall observe that logical and structural composition can be fruitfully composed. For example, any structural composition can be followed by a logical composition to synchronize on sets of atomic propositions. Finally, structural composition refines logical composition, but the reverse does not hold.

Example 4. *This example illustrates how synchronization on a parallel composition of* S *and* S' *in Fig. 5a can be achieved, by conjoining the parallel composition in Fig. 5b with a synchronizer* Sync *(Fig. 5c). This particular synchronizers removes the valuations from* $S \parallel S'$ *that do not satisfy* $(a = d) \wedge (b = \neg c)$*, giving rise to the CMC in Fig. 5d.*

An inconsistency appears in state $(1, 1)$ *of* $(S \parallel S') \wedge$ Sync *meaning that this CMC is inconsistent and there is no implementation of the two CMCs that obey the synchronizer.*

5 Abstraction

As any existing formal technique, CMCs may suffer from the so called state-space explosion problem. A solution to this problem is the one of abstraction. The technique aims at model reduction by collapsing sets of concrete states to abstract states. Here,



(a) Two CMCs S and S'

(b) $S \parallel S'$

(c) Synchronizer Sync

(d) $(S \parallel S') \wedge$ Sync

Figure 5: Parallel composition and synchronization of CMCs.

we propose to conduct such an abstraction by partitioning the set of concrete states into a set of smaller size. We will also propose another abstraction that permits to replace a CMC by an IMC.

Let us introduce an example that will be used through the rest of the section.

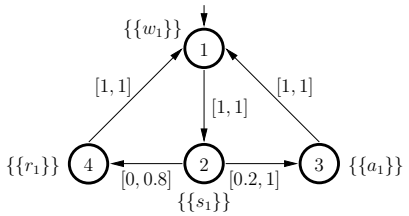
Example 5. *Consider four researchers that behave identically. In the initial state, Researcher i writes a paper. This is represented with the atomic proposition* w_i*. Then, with probability* 1*, the researcher sends the paper to a conference, represented with the atomic proposition* s_i*. The paper is accepted with a probability greater than* 20% *(atomic proposition* a_i*), and rejected with a probability below* 80% *(*r_i*). In both cases, the researcher goes back to writing. Researcher i is represented with the CMC* $S_i = \langle\{1,2,3,4\}, 1, \varphi_i, \{w_i, s_i, a_i, r_i\}, V_i\rangle$*.*

The model of Researcher 1 is given in Fig. 6a and Researcher 2 is specified in Fig. 6b. Since the models are independent (their sets of atomic propositions are disjoint), we can compute their parallel composition S_{12}*, shown in Fig. 7.*

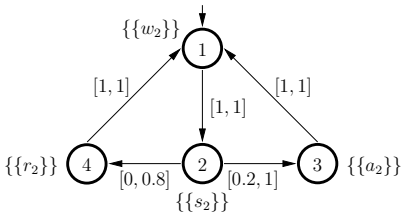
5.1 State-based Abstraction

We consider an abstraction function that works by abstracting the set of states. A state abstraction function is a surjection $\alpha : \{1, \ldots, k\} \to \{1, \ldots, k'\}$ for some $k' \leq k$, such that $\{1, \ldots, k\} = \bigcup_{i' \in \{1,\ldots,k'\}} \alpha^{-1}(i')$ (totality).

The state abstraction of a distribution μ over $\{1, \ldots, k\}$, denoted $\alpha(\mu) \in \mathit{Dist}(\{1, \ldots, k'\})$, is defined as $\alpha(\mu)(i') = \mu(\alpha^{-1}(i')) = \sum_{i \in \alpha^{-1}(i')} \mu(i)$ for all $i' \in \{1, \ldots, k'\}$.

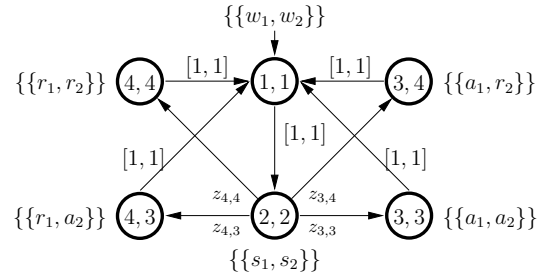


(a) CMC S_1 representing Researcher 1.



(b) CMC S_2 representing Researcher 2.

Figure 6: Example of two researchers.



$$\varphi(2,2)(z) \equiv \exists x, y,\ z_{1,1} = x_1 y_1 \wedge z_{1,2} = x_1 y_2 \ldots$$
$$\wedge x_1 = 0 \wedge x_2 = 0 \wedge x_3 \in [0.2, 1] \wedge x_4 \in [0, 0.8] \wedge x_3 + x_4 = 1$$
$$\wedge y_1 = 0 \wedge y_2 = 0 \wedge y_3 \in [0.2, 1] \wedge y_4 \in [0, 0.8] \wedge y_3 + y_4 = 1$$

Figure 7: CMC S_{12} representing the independent parallel composition of Researchers 1 and 2.

Definition 13 (State abstraction)**.** *Let $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ be a CMC and let $\alpha : \{1, \ldots, k\} \to \{1, \ldots, k'\}$ be a state abstraction function. The CMC $\alpha(S) = \langle \{1, \ldots, k'\}, \alpha(o_S), \varphi', A_S, V'_S \rangle$ is induced by α such that*

$$\varphi'(i')(y_1, \ldots, y_{k'}) \equiv \exists x_1, \ldots, x_k \in [0, 1] :$$
$$(y_1, \ldots, y_{k'}) = \alpha((x_1, \ldots, x_k)) \wedge \bigvee_{i \in \alpha^{-1}(i')} \varphi(i)(x_1, \ldots, x_k),\ \textit{and}$$

$$V'_S(i') = \bigcup_{i \in \alpha^{-1}(i')} V_S(i).$$

The following theorem shows that the above construction is indeed an abstraction with respect to refinement.

Theorem 5. *Let S be a CMC. It holds that $S \preceq \alpha(S)$.*

Proof. Let $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ be a CMC, let $\alpha : \{1, \ldots, k\} \to \{1, \ldots, k'\}$ be a state abstraction function, and let $\alpha(S) = \langle \{1, \ldots, k'\}, \alpha(o_S), \varphi', A_S, V'_S \rangle$ be the CMC induced by α.

We define a relation $\mathcal{R} \subseteq \{1, \ldots, k\} \times \{1, \ldots, k'\}$ such that $u \mathcal{R} v \iff \alpha(u) = v$. We show that $\mathcal{R}$ is a weak refinement relation.

1. By definition, we have $V_S(u) \subseteq \bigcup_{i \in \alpha^{-1}(v)} V_S(i) = V'_S(v)$.

2. Construct the matrix $\Delta \in [0,1]^{k \times k'}$ as $\Delta_{ii'} = 1$ if $\alpha(i) = i'$ and 0 otherwise. By construction, this is a correspondence matrix. Let $x \in [0,1]^k$ be such that $\varphi(u)(x)$ holds.

   - Let $i \in \{1, \ldots, k\}$ be such that $x_i \neq 0$. We have that $\alpha(i) = i'$ for exactly one $i' \in \{1, \ldots, k'\}$, so $\sum_{j=1}^{k'} \Delta_{ij} = 1$.
   - Let $i \in \{1, \ldots, k'\}$. The i'th entry of $x\Delta$ is computed as

   $$\begin{aligned}[x\Delta]_i &= \sum_{j=1}^{k} x_j \Delta_{ji} = \sum_{j:\alpha(j)=i} x_j \\ &= \sum_{j \in \alpha^{-1}(i)} x_j,\end{aligned}$$

   so $\alpha(x) = x\Delta$. By this fact, $\varphi'(v)(x\Delta)$.
   - Assume that for $u' \in \{1, \ldots, k\}$ and $v' \in \{1, \ldots, k'\}$ that $\Delta_{u'v'} \neq 0$. By definition $\alpha(u') = v'$ and therefore $u' \mathcal{R} v'$.

Finally, since $o_S \mathcal{R} \alpha(o_S)$, we conclude that $\mathcal{R}$ is a weak refinement relation. □

Example 6. *Continuing Example 5, we consider the two researchers and their composition that are given in Figures 6 and 7, respectively. We consider the case where one is only interested in the acceptance of at least one paper. In order to avoid state-space explosion when composing with other CMCs, we suggest to group states that represent the acceptance of at least one paper, i.e. States* $(3,3)$*,* $(3,4)$ *and* $(4,3)$ *in CMC* S_{12}*. This is done with the following state abstraction function:*

$$\alpha : \begin{cases} (1,1) \mapsto 1' \\ (2,2) \mapsto 2' \\ (3,3) \mapsto 3' \\ (3,4) \mapsto 3' \\ (4,3) \mapsto 3' \\ (4,4) \mapsto 4' \end{cases}$$

CMC $\alpha(S_{12})$ *is given in Figure 8, where State* $3'$ *is an abstraction for all the states of* S_{12} *where at least* 1 *paper is accepted.*

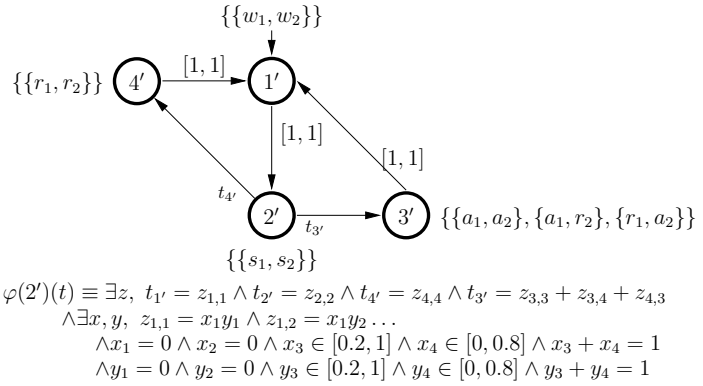


Figure 8: CMC $\alpha(S_{12})$ representing the state-abstraction of CMC S_{12}.

State-abstractions can be composed as follows: Let $\alpha_1 : Q_1 \to Q_1'$ and $\alpha_2 : Q_2 \to Q_2'$ be two state abstractions, we define the composition of α_1 and α_2 as the state-abstraction $\alpha_1 \times \alpha_2 : Q_1 \times Q_2 \to Q_1' \times Q_2'$ such that for all $q_1 \in Q_1$ and $q_2 \in Q_2$, $(\alpha_1 \times \alpha_2)(q_1, q_2) = (\alpha_1(q_1), \alpha_2(q_2))$. The following theorem states that abstraction is compositional.

Theorem 6. *Let* $S_1 = \langle \{1, \ldots, k_1\},\ o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_1 \cap A_2 = \emptyset$*, and let* $\alpha_1 : \{1, \ldots, k_1\} \to \{1, \ldots, k_1'\}$ *and* $\alpha_2 : \{1, \ldots, k_2\} \to \{1, \ldots, k_2'\}$ *be state abstraction functions. It holds that* $\alpha_1(S_1) \| \alpha_2(S_2) = (\alpha_1 \times \alpha_2)(S_1 \| S_2)$ *up to isomorphism.*

Proof. Let $S_1 = \langle \{1, \ldots, k_1\},\ o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be CMCs with $A_1 \cap A_2 = \emptyset$, and let $\alpha_1 : \{1, \ldots, k_1\} \to \{1, \ldots, k_1'\}$ and $\alpha_2 : \{1, \ldots, k_2\} \to \{1, \ldots, k_2'\}$ be state abstraction functions.

We build CMCs $\alpha_1(S_1) \| \alpha_2(S_2)$ and $(\alpha_1 \times \alpha_2)(S_1 \| S_2)$, and show that they are syntactically equivalent.

- $\alpha_1(S_1) \| \alpha_2(S_2) = \langle \{1, \ldots, k_1'\} \times \{1, \ldots, k_2'\}, (\alpha_1(o_1), \alpha_2(o_2)), \varphi, A_1 \cup A_2, V \rangle$, with
  - $\varphi((i', j'))(z) = 1$ iff there exists $x' \in [0,1]^{k_1'}$ and $y' \in [0,1]^{k_2'}$ such that $z_{(i'', j'')} = x'_{i''} y'_{j''}$ for all i'', j'', and there exists $x \in [0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $x' = \alpha_1(x)$, $y' = \alpha_2(y)$ and there exists $i \in \alpha_1^{-1}(i')$ and $j \in \alpha_2^{-1}(j')$ such that $\varphi_1(i)(x) = \varphi_2(j)(y) = 1$.
  - $V((i', j')) = \{Q_1 \cup Q_2 \mid Q_1 \in \cup_{i \in \alpha_1^{-1}(i')} V_1(i)$ and $Q_2 \in \cup_{j \in \alpha_2^{-1}(j')} V_2(j)\}$.
- $(\alpha_1 \times \alpha_2)(S_1 \| S_2) = \langle \{1, \ldots, k_1'\} \times \{1, \ldots, k_2'\}, (\alpha_1(o_1), \alpha_2(o_2)), \varphi', A_1 \cup A_2, V' \rangle$, with
  - $\varphi'((i', j'))(z') = 1$ iff there exists $z \in [0,1]^{k_1 \times k_2}$ such that $z' = (\alpha_1 \times \alpha_2)(z)$ and there exists $(i, j) \in (\alpha_1 \times \alpha_2)^{-1}((i', j'))$ such that there exists $x \in$

$[0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $z_{i'',j''} = x_{i''}y_{j''}$ for all i'', j'' and $\varphi_1(i)(x) = \varphi_2(j)(y) = 1$.

- $V'((i',j')) = \cup_{(i,j)\in(\alpha_1\times\alpha_2)^{-1}((i',j'))}\{Q_1 \times Q_2 \mid Q_1 \in V_1(i) \text{ and } Q_2 \in V_2(j)\}$.

Since, by construction, $(\alpha_1 \times \alpha_2)^{-1}(i',j') = \alpha_1^{-1}(i) \times \alpha_2^{-1}(j)$, both the constraint functions φ and φ' and the valuation functions V and V' are equivalent.

□

5.2 From CMCs to IMCs

One of the problems of CMCs is that the constraints obtained after composition may be too complex to be efficiently handled by tools. A solution is to abstract those constraints with intervals. This is done by an abstraction χ that builds an IMC $\chi(S)$ from a given CMC S. Unlike the state abstraction, the constraint abstraction does not merge states, but it simplifies the probability constraints.

Definition 14 (Constraint abstraction)**.** *Let $S = \langle\{1,\dots,k\}, o_S, \varphi, A_S, V_S\rangle$ be a CMC and let $C \subseteq \{1,\dots,k\}$. The constraint-abstracted CMC $\chi(S) = \langle\{1,\dots,k\}, o_S, \varphi', A_S, V_S\rangle$ is defined such that for all $1 \le i \le k$ and $y \in [0,1]^k$,*

$$\varphi'(i)(y_1,\dots,y_k) \equiv \bigwedge_{j=1}^{k} y_j \in I_j^i \wedge \sum_{j=1}^{k} y_j = 1,$$

where $I_1^i,\dots,I_k^i$ are the smallest closed intervals such that $\forall x \in [0,1]^k, \varphi(i)(x) \Rightarrow \bigwedge_{j=1}^k x_j \in I_j^i$.

We now show that the constraint abstraction presented above is indeed an abstraction, i.e. for all CMC S, we have $S \preceq \chi(S)$.

Theorem 7. *Let $S = \langle\{1,\dots,k\}, o_S, \varphi, A_S, V_S\rangle$ be a CMC. Then $S \preceq \chi(S)$.*

Proof. We define a relation $\mathcal{R} \subseteq \{1,\dots,k\} \times \{1,\dots,k\}$ such that $u\,\mathcal{R}\,v \iff u = v$. Consider such u and v. We show that $\mathcal{R}$ is a weak refinement relation.

1. By definition, we have $V_S(u) \subseteq V_S(v)$.
2. Construct the matrix $\Delta \in [0,1]^{k\times k}$ as $\Delta_{ii'} = 1$ if $i = i'$ and 0 otherwise. Observe that Δ is a correspondence matrix. Let $x \in [0,1]^k$ such that $\varphi(u)(x)$.
   - Let $i \in \{1,\dots,k\}$ be such that $x_i \neq 0$. We have that $i = i'$ for exactly one $i' \in \{1,\dots,k\}$, so $\sum_{j=1}^k \Delta_{ij} = 1$.
   - Since Δ is the identity matrix, it holds that $x\Delta = x$. Let $1 \le i \le k$. By definition, whenever $\varphi(u)(x)$ holds, we have $x_i \in I_i^u$, and thus $x_i \in I_i^v$. As a consequence, $\forall 1 \le i \le k$, we have $x_i \in I_i^v$. Thus $\varphi'(v)(x)$ holds.
   - Assume that for $u' \in \{1,\dots,k\}$ and $v' \in \{1,\dots,k\}$ that $\Delta_{u'v'} \neq 0$. By definition $u' = v'$ and therefore $u'\,\mathcal{R}\,v'$.

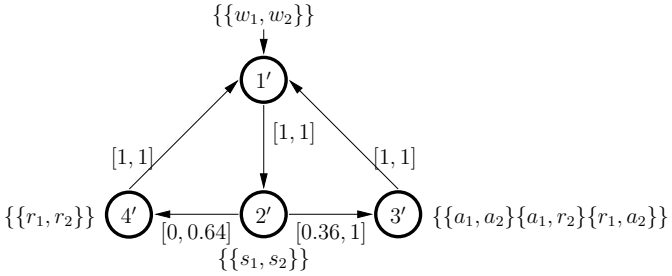


Figure 9: The constraint-abstraction $\chi(\alpha(S_{12}))$ of CMC $\alpha(S_{12})$.

Finally, since $o_S \mathcal{R} o_S$, we conclude that $\mathcal{R}$ is a weak refinement relation. □

Example 7. *Continuing our running example, we observe that the constraints of CMC $\alpha(S)$ in Figure 8 are quite complex. We propose to use constraint-abstraction in order to produce a CMC $\chi(\alpha(S_{12}))$ with simpler constraints using intervals. Such CMC is given in Figure 9. Observe that the new intervals obtained in $\chi(\alpha(S_{12}))$ ensure that the probability of having at least one paper accepted will be greater than 36%.*

We now show that χ characterizes the smallest IMC in single valuation normal form that abstracts a deterministic CMC in single valuation normal form. Observe that this results does not hold in general for non deterministic CMCs.

Theorem 8. *Let $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ be a CMC. If S is deterministic and in single valuation normal form, then $\chi(S)$ is the smallest IMC in single valuation normal form abstracting S, i.e. for all IMCs S' in single valuation normal form, such that $S \preceq S'$, it holds that $\chi(S) \preceq S'$.*

Proof. Let $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ be a CMC and let $S' = \langle \{1, \ldots, m\}, o_{S'}, \varphi_{S'}, A_{S'}, V_{S'} \rangle$ be an IMC, both in single valuation normal form. Assume that S is deterministic and that $S \preceq S'$ holds. Let $\mathcal{R} \subseteq \{1, \ldots, k\} \times \{1, \ldots, m\}$ be the weak refinement relation witnessing this. Let $\chi(S) = \langle \{1, \ldots, k\}, o_S, \varphi', A_S, V'_S \rangle$.

Define the relation $\mathcal{R}' := \mathcal{R}$. We show that $\mathcal{R}'$ is a weak refinement relation between $\chi(S)$ and S'.

Let $u \in \{1, \ldots, k\}$ and $v \in \{1, \ldots, m\}$ be such that $u \mathcal{R} v$. Let $1 \leq i \leq k$ and K_i be the set of states j of S' such that $i \mathcal{R}' j$. Formally, $K_i = \{j \in \{1, \ldots, m\} \mid i \mathcal{R}' j\}$. Observe that since S is deterministic and S' is in single valuation normal form, we have $K_i \cap K_{i'} = \emptyset$ for all $i \neq i'$ such that there exist x and $y \in [0,1]^k$ such that $\varphi(u)(x) = \varphi(u)(y) = 1$ and $x_i > 0$ and $y_{i'} > 0$ (A).

Let $S'^u_i = [l^u_i, u^u_i]$, $1 \leq i \leq k$ be the intervals associated to u in $\chi(S)$ and let $[m^v_j, M^v_j]$, $1 \leq j \leq m$ be the intervals associated to v in S'.

Let $1 \leq i \leq k$ and let $x^{l,i} \in [0,1]^k$ be such that $\varphi(u)(x^{l,i})$ holds and $x^{l,i}_i = l^u_i$. Such an $x^{l,i}$ exists because of the definition of $\chi(S)$. Since $u \mathcal{R} v$, there exists a correspondence

matrix $\Delta^{l,i}$ such that $\varphi_{S'}(v)(x^{l,i}\Delta^{l,i})$ holds. As a consequence, we have that

$$\forall j \notin K_i,\ \Delta^{l,i}_{i,j} = 0 \tag{5}$$

$$\forall i' \neq i, \forall j \in K_i,\ \Delta^{l,i}_{i'j} = 0 \tag{6}$$

By $\mathcal{R}$, we have that for all $1 \leq i' \leq k$, $\sum_{1\leq j\leq m} x^{l,i}_{i'}\Delta^{l,i}_{i',j} = x^{l,i}_{i'}$. In particular, for $i' = i$, we have that $\sum_{1\leq j\leq m} x^{l,i}_{i}\Delta^{l,i}_{i,j} = x^{l,i}_{i} = l^u_i$. Thus, by (5), we have that $\sum_{j\in K_i} x^{l,i}_{i}\Delta^{l,i}_{i,j} = l^u_i$.

Moreover, since $\varphi_{S'}(x^{l,i}\Delta^{l,i})$ holds, we have that for all $1 \leq j \leq m$, $[x^{l,i}\Delta^{l,i}]_j \geq m^v_j$. Thus, $\sum_{1\leq i'\leq k} x^{l,i}_{i'}\Delta^{l,i}_{i',j} \geq m^v_j$. By (6), we thus obtain that for all $j \in K_i$, $\sum_{1\leq i'\leq k} x^{l,i}_{i'}\Delta^{l,i}_{i',j} = x_i\Delta^{l,i}_{i,j} \geq m^v_j$. As a consequence, we have

$$l^u_i \geq \sum_{j\in K_i} m^v_j.$$

Similarly, we obtain that for all $1 \leq i \leq k$,

$$\sum_{j\in K_i} m^v_j \leq l^u_i \quad \text{and} \quad u^u_i \leq \sum_{j\in K_i} M^v_j.$$

We now show that $\mathcal{R}'$ is a weak refinement relation. Let $u \in \{1,\ldots,k\}$ and $v \in \{1,\ldots,m\}$ be such that $u\,\mathcal{R}'\,v$.

1. Since $u\,\mathcal{R}\,v$, we have that $V_S(u) \subseteq V_S(v)$. By definition of $\chi(S)$, we thus have $V'_S(u) \subseteq V_S(v)$.
2. Let $x \in [0,1]^k$ be such that $\varphi'(u)(x)$ holds, i.e. $\forall 1 \leq i \leq k,\ l^u_i \leq x_i \leq u^u_i$. We define a correspondence matrix Δ' as follows: for all $1 \leq i \leq k$ such that $x_i = 0$, let $\Delta'_{i,j} = 0$ for all j. Otherwise, define

$$\Delta'_{i,j} = \begin{cases} \frac{1}{x_i}\left(m^v_j + \frac{(M^v_j - m^v_j)(x_i - \sum_{j'\in K_i} m^v_{j'})}{\sum_{j'\in K_i}(M^v_{j'} - m^v_{j'})}\right) & \text{if } j \in K_i \\ 0 & \text{otherwise} \end{cases}$$

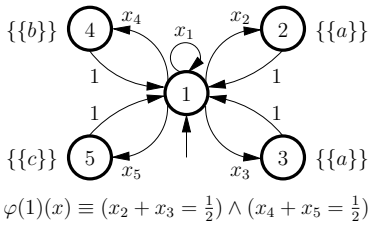
   By definition, we have that whenever $x_i > 0$, $\sum_{j=1}^m \Delta'_{i,j} = 1$. Let $1 \leq j \leq m$. By the observation (A) above, there exists at most one $1 \leq i^j \leq k$ such that $x_{i^j} > 0$ and $j \in K_i$. Consider $y = x\Delta'$. If there is no i^j such that $x_{i^j} > 0$, then it holds that $m^v_j = 0$ and $m^v_j \leq y_j \leq M^v_j$.

   Otherwise, by definition,

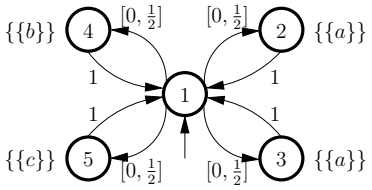
$$y_j = \sum_{i=1}^k x_i\Delta'_{i,j} = x_{i^j}\Delta'_{i^j,j} = m^v_j + \frac{(M^v_j - m^v_j)(x_{i^j} - \sum_{j'\in K_{i^j}} m^v_{j'})}{\sum_{j'\in K_{i^j}}(M^v_{j'} - m^v_{j'})}.$$

   Since $\sum_{j'\in K_{i^j}} m^v_{j'} \leq l^u_{i^j} \leq x_{i^j}$, we have $y_j \geq m^v_j$. Similarly, since $x_{i^j} \leq u^u_{i^j} \leq \sum_{j'\in K_{i^j}} M^v_{j'}$, we have $y_j \leq M^v_j$.

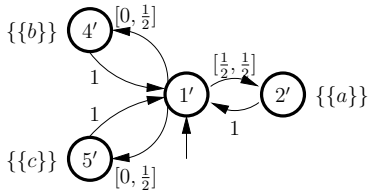
   As a consequence, $\varphi_{S'}(v)(y)$ holds.



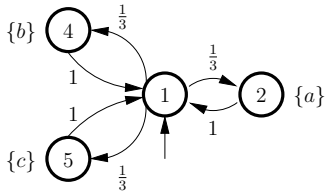
(a) A non-deterministic CMC S.



(b) The constraint abstraction $\chi(S)$ of CMC S.



(c) IMC S' such that $S \preceq S'$ and $\chi(S) \not\preceq S'$.



(d) A MC I such that $I \models \chi(S)$ and $I \not\models S'$.

Figure 10: A counter-example for Thm. 8 in the non-deterministic case

3. Finally, whenever $\Delta'_{i,j} > 0$, we have $j \in K_i$ and as a consequence, $i \mathcal{R}' j$.

Since $o_S \mathcal{R} o_{S'}$, we have $o_S \mathcal{R}' o_{S'}$ and we conclude that $\mathcal{R}'$ is a weak refinement relation. □

The above theorem holds for the case where CMC S is both deterministic and in single valuation normal form. Otherwise $\chi(S)$ may not be the smallest IMC abstracting S. Consider applying χ to the nondeterministic CMC $S = \langle \{1, 2, 3, 4, 5\}, 1, \varphi, \{a, b, c\}, V \rangle$ given in Figure 10a. The result is the IMC $\chi(S)$ given in Figure 10b. Consider now IMC S', given in Figure 10c. S' abstracts S by merging states 2 and 3 into a single state $2'$. It is easy to see that $S \preceq S'$. However, $\chi(S)$ is not a refinement for S'. Indeed, MC I, given in Figure 10d, is an implementation of $\chi(S)$ but not an implementation of S'. As a consequence, $\chi(S)$ is not the smallest IMC abstracting S.

Contrary to state abstraction, constraint abstraction is not fully compositional. This is not surprising as the composition of two Interval Markov Chains is not an Interval Markov Chain. Formally:

Proposition 9. *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be two CMCs with* $A_1 \cap A_2 = \emptyset$. *We have* $\chi(S_1) \parallel \chi(S_2) \preceq \chi(S_1 \parallel S_2)$.

Proof. Let $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be two CMCs with $A_1 \cap A_2 = \emptyset$. Let $S_1 \parallel S_2 = \langle \{1,\dots,k_1\} \times \{1,\dots,k_2\}, (o_1,o_2), \varphi_\parallel, A_1 \cup A_2, V_\parallel\rangle$ be their parallel composition, and let φ_1^χ, φ_2^χ, $\varphi_\parallel^\chi$ and φ^χ denote respectively the constraints of $\chi(S_1)$, $\chi(S_2)$, $\chi(S_1 \parallel S_2)$ and $\chi(S_1) \parallel \chi(S_2)$. By construction, it holds that $(\chi(S_1) \parallel \chi(S_2))$ and $\chi(S_1 \parallel S_2)$ have the exact same state-space, initial states, atomic propositions and valuations. We now show that whatever solution of φ^χ is a solution of $\varphi_\parallel^\chi$.

Let $(I_{i'}^i)_{i' \in \{1,\dots,k_1\}}$ be the intervals associated to state i in $\chi(S_1)$. Similarly, let $(J_{j'}^j)_{j' \in \{1,\dots,k_2\}}$ and $(L_{(i',j')}^{(i,j)})_{(i',j') \in \{1,\dots,k_1\}\times\{1,\dots,k_2\}}$ be the intervals associated respectively to state j in $\chi(S_2)$ and to state (i,j) in $\chi(S_1 \parallel S_2)$. Additionally, let $I_{i'}^i = [m_{i'}^{I,i}, M_{i'}^{I,i}]$, let $J_{j'}^j = [m_{j'}^{J,j}, M_{j'}^{J,j}]$ and let $L_{(i',j')}^{(i,j)} = [m_{(i',j')}^{L,(i,j)}, M_{(i',j')}^{L,(i,j)}]$.

Consider two states $i \in \{1,\dots,k_1\}$ and $j \in \{1,\dots,k_2\}$. By contradiction, we show that for all transition vectors $x \in [0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $\varphi_1^\chi(i)(x) = \varphi_2^\chi(y) = 1$, it holds that $\varphi_\parallel^\chi((i,j))(z) = 1$, with $z \in [0,1]^{k_1 \times k_2}$ such that $z_{(i',j')} = x_{i'} y_{j'}$.

Suppose that there exists $x \in [0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $\varphi_1^\chi(i)(x) = \varphi_2^\chi(y) = 1$ and $\varphi_\parallel^\chi((i,j))(z) \neq 1$, with $z \in [0,1]^{k_1 \times k_2}$ such that $z_{(i',j')} = x_{i'} y_{j'}$. As a consequence, there must exist states $i' \in \{1,\dots,k_1\}$ and $j' \in \{1,\dots,k_2\}$ such that $x_{i'} y_{j'} \notin L_{(i',j')}^{(i,j)}$, thus either $x_{i'} y_{j'} > M_{(i',j')}^{L,(i,j)}$, or $x_{i'} y_{j'} < m_{(i',j')}^{L,(i,j)}$.

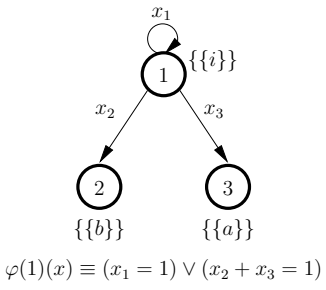
Suppose that the former holds (the second case being similar). By the minimality and convexity of $I_{i'}^i$ and $J_{j'}^j$, for all constant $\epsilon > 0$, there must exist $x' \in [0,1]^{k_1}$ and $y' \in [0,1]^{k_2}$ such that $\varphi_1(i)(x') = \varphi_2(j)(y') = 1$ and $(x_{i'} - x'_{i'}) < \epsilon$ and $(y_{j'} - y'_{j'}) < \epsilon$. Consider $\epsilon = \frac{(x_{i'} y_{j'} - M_{(i',j')}^{L,(i,j)})}{2}$. It then holds that $x'_{i'} y'_{j'} > M_{(i',j')}^{L,(i,j)}$. However, by hypothesis, we have that the transition vector $z' \in [0,1]^{k_1 \times k_2}$ such that $z'_{(i',j')} = x'_{i'} y'_{j'}$ satisfies $\varphi_\parallel((i,j))$. This contradicts the definition of $L_{(i',j')}^{(i,j)}$.

As a consequence, we have that for all transition vectors $x \in [0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $\varphi_1^\chi(i)(x) = \varphi_2^\chi(y) = 1$, it holds that $\varphi_\parallel^\chi((i,j))(z) = 1$, with $z \in [0,1]^{k_1 \times k_2}$ such that $z_{(i',j')} = x_{i'} y_{j'}$. Thus the identity relation is a refinement relation between $(\chi(S_1) \parallel \chi(S_2))$ and $\chi(S_1 \parallel S_2)$.

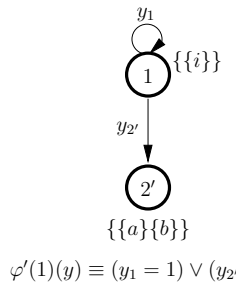
□

State abstraction and constraint abstraction cannot be compared. Consider CMC S given in Figures 11a. Consider the state abstraction α grouping states 2 and 3 of S. The result of applying this abstraction to S, $\alpha(S)$ is given in Figure 11b. The constraint abstraction of S, $\chi(S)$, is given in Figure 11c. It is obvious that there is no refinement relation between $\alpha(S)$ and $\chi(S)$. State $2'$ of $\alpha(S)$ cannot refine states 2 or 3 of $\chi(S)$ as they disagree on valuations. On the other hand, state 1 of $\chi(S)$ cannot refine state 1 of $\alpha(S)$ due to its constraints.

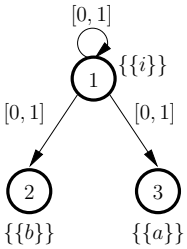
Example 8. *Figure 12 summarizes the composition/abstraction process computing a single CMC with interval constraints abstracting the composition of 4 independent re-*



(a) A CMC S.



(b) State abstraction $\alpha(S)$ of CMC S



(c) Constraint abstraction $\chi(S)$ of CMC S

Figure 11: The two abstractions produce incomparable results

searchers. Observe that, without abstraction, the resulting CMC would have approximately 4^4 *states.*

6 Implementation of The APAC Tool

APAC is an implementation of the specification theory based on CMCs. APAC can also handle the more recent formalism of Abstract Probabilistic Automata [64] that is a specification theory for probabilistic automata [6].

6.1 APAC: introduction and functionality

The APAC tool is implemented in C# using the Z3 [112] SMT solver developed in Microsoft Research. We exploit the quantifier elimination algorithms for linear arithmetics over real numbers implemented in Z3. Furthermore, we use the ANTLR Parser Generator for parsing the input files, as well as functionality for storing abstract syntax trees for constraints. The tool is freely available at `http://www.cs.aau.dk/~mikkelp/apac`, including documentation.

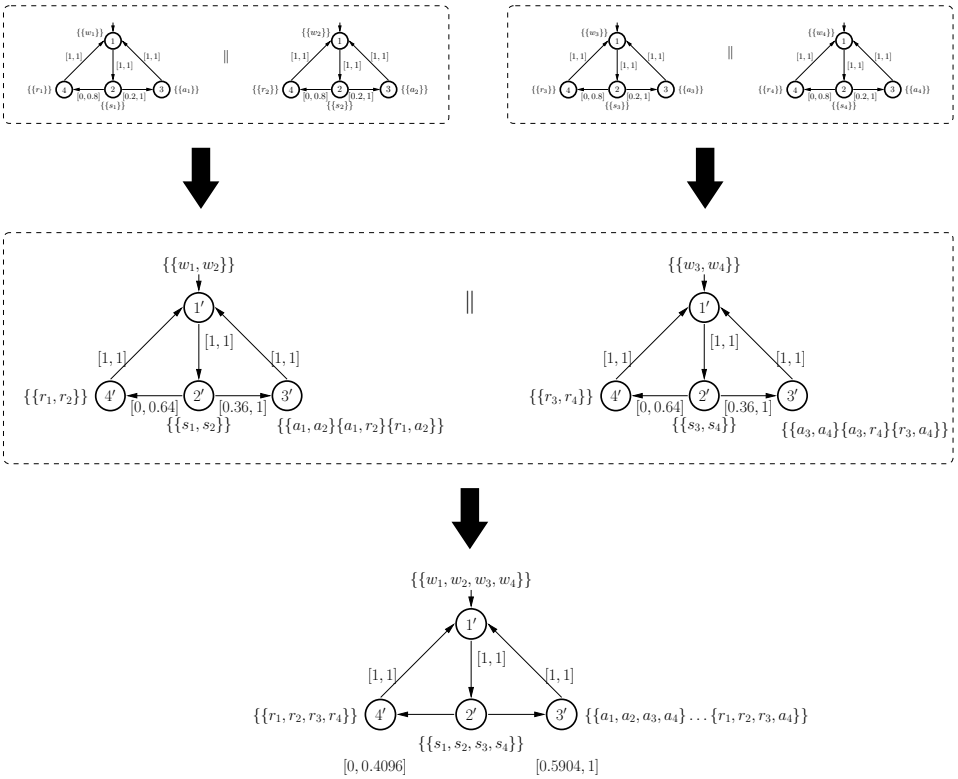


Figure 12: Composition-abstraction process for the composition of 4 researchers.

The input given to APAC is a text file containing one or more definitions of CMCs followed by a statement specifying the operations to be evaluated:

```
Name: S1;
AP:(l,m,n,o);
state 1:((l)): x[1]=0.0 && x[2]+x[3]>=7/10 && x[3]+x[4]>=2/10;
state 2:((m)): x[2]=1.0;
state 3:((n)): x[3]=1.0;
state 4:((o)): x[4]=1.0;

Name: S2;
AP:(l,m,n,o);
state 1:((l)): x[1]=0.0 && x[2]+x[3]>=7/10 && x[4]+x[5]>=2/10;
state 2:((m)): x[2]=1.0;
state 3:((n)): x[3]=1.0;
state 4:((n)): x[4]=1.0;
state 5:((o)): x[5]=1.0;

check: S1 wref S2; show S1; D(S1);
```

The above example defines CMCs S_1 and S_2 and, in the final line, requests checking existence of a weak refinement relation between S_1 and S_2. Then S_1 should be printed to the console, and we check whether S_1 is deterministic.

All interaction with APAC is done through such statements. Table 3 defines the syntax of the available operators. Variables not mentioned in constraints are free. For example, in state 2 of S_1, variables $x[1]$, $x[3]$, and $x[4]$ are free. In this case, they are effectively forced to equal zero, since all variables in the same probability distribution need to sum up to 1.

The `<set definition>` (see Table 3) defines how the state space is partitioned for the purpose of a state abstraction. For instance, for a CMC S_1 with states $\{1, 2, 3, 4, 5\}$, one valid set definition is `(1,2)(3,4,5)`, which leads to the result of the abstraction having 2 states; one state corresponding to the grouping of 1 and 2, and one for the grouping of 3, 4, and 5, respectively.

6.2 Algorithms in APAC

In the following we discuss the encoding of some operators in Z3. All the operations that are not discussed here can be easily implemented by following their definitions; with an exception for structural composition. This operation involves multiplication and cannot be handled with Z3.

Refinement We first present a different, but equivalent, definition of weak refinement. This is needed as the classical definition of weak refinement involves multiplication, which is not allowed by Z3.

Definition 15 (Weak Refinement)**.** *Let* $S_1 = \langle\{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ *and* $S_2 = \langle\{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ *be CMCs with* $A_2 \subseteq A_1$*. The relation* $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ *is a* weak refinement relation *iff* $v \mathcal{R} u$ *implies:*

1. $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$ *and*
2. *For any distribution* $x \in [0, 1]^{k_1}$ *satisfying* $\varphi_1(v)(x)$*, there exists a matrix* $\Delta \in [0, 1]^{k_1 \times k_2}$ *such that*
   - *For all* S_1 *states* $1 \leq i \leq k_1$*,* $x_i = \sum_{j=1}^{k_2} \Delta_{ij}$*;*
   - $\varphi_2(u)\left(\sum_{j=1}^{k_1} \Delta_{i1}, \ldots, \sum_{j=1}^{k_1} \Delta_{ik_2}\right)$ *holds and*
   - $\Delta_{v'u'} \neq 0 \Rightarrow v' \mathcal{R} u'$*.*

CMC S_1 *(weakly) refines* S_2*, written* $S_1 \preceq S_2$*, iff* $o_1 \mathcal{R} o_2$*.*

The computations proceeds by a coinductive iteration until a fixpoint is reached. See Algorithm 1. The outer loop continues as long as changes in the relation are performed (line 5). In each iteration the remaining pairs are considered, and a pair is removed if it violates the requirement on valuations or the requirement on redistribution.

The requirement on valuations in line 3 is handled without using Z3. In lines 8–11 we deliver to Z3 an encoding of the correspondence condition for the probability distributions in the definitions of weak refinement. Since the corresponding formula is

a sentence (has no free variables), the quantifier elimination algorithm will evaluate the formula to a value of true or false.

If it turns out that the refinement between S_1 and S_2 does not hold, and S_1 and S_2 are deterministic, consistent, and are in single valuation normal form, APAC provides a counterexample. The counterexample is a Markov chain P such that $P \models S_1$ and $P \not\models S_2$. Since under these conditions $S_1 \not\preceq S_2$ is equivalent to $[\![S_1]\!] \not\subseteq [\![S_2]\!]$, such a Markov chain is guaranteed to exists.

The algorithm for the counterexample generation relies on the following lemma, which is a direct consequence of determinism and single valuation normal form [62]:

Lemma 10 ([62]). *Let $S = \langle \{1,\dots,k\}, o, \varphi, A, V\rangle$ and $S' = \langle \{1,\dots,k'\}, o', \varphi', A', V'\rangle$ be deterministic CMCs in single valuation normal form such that $S \not\preceq S'$. Let $(i,j) \in \{1,\dots,k\} \times \{1,\dots,k'\}$. For all $i' \in \{1,\dots,k\}$, there exists at most one $j' \in \{1,\dots,k'\}$ such that $V(i') = V'(j')$ and there exists a distribution $y \in [0,1]^{k'}$ such that $\varphi'(j)(y)$ and $y_{j'} > 0$.*

In what follows, we use $\mathsf{succ}_{(i,j)}(i')$ to define the unique state j' introduced in the previous lemma, if it exists, and $\perp$ otherwise. During refinement checking, the tool monitors, for each pair (i,j) if

1. (i,j) was removed from $\mathcal{R}$ because of a disagreement on sets of valuations,
2. (i,j) was removed from $\mathcal{R}$ because of a non-redistributable distribution π,

Algorithm 1: Checking Weak Refinement

Input : CMCs $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and
$S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be CMCs with $A_2 \subseteq A_1$
Output: A weak refinement relation $\mathcal{R}$ that may contain (o_1, o_2)

$\mathcal{R} = \{1,\dots,k_1\} \times \{1,\dots,k_2\}$;
foreach $(i,j) \in \mathcal{R}$ **do**
  if $V_1(i)\downarrow_{A_2} \not\subseteq V_2(j)$ **then**
    remove (i,j) from $\mathcal{R}$;
repeat
  changed = false;
  foreach $(i,j) \in \mathcal{R}$ **do**
    if $\neg(\forall x \in [0,1]^{k_1} : \varphi_1(i)(x) \Rightarrow \exists \Delta \in [0,1]^{k_1 \times k_2} :$
      $\varphi_2(j)\left(\sum_{i=1}^{k_1} \Delta_{i1}, \dots, \sum_{i=1}^{k_1} \Delta_{ik_2}\right) \wedge$
      $\forall 1 \le i' \le k_1 : x_i = \sum_{j'=1}^{k_2} \Delta_{i'j'} \wedge$
      $\forall 1 \le i' \le k_1, \forall 1 \le j' \le k_1 : \Delta_{i'j'} \neq 0 \Rightarrow i' \,\mathcal{R}\, j')$ **then**
      changed = true;
      remove (i,j) from $\mathcal{R}$;
until *not* changed;

3. or (i, j) is still in the relation.

Let $P = \langle Q, o, M, A, V \rangle$ be defined with

- $Q = \{1, \ldots, k_1\} \times (\{1, \ldots, k_2\} \cup \{\bot\})$,
- $o = (o_1, o_2)$,
- For all $(i, j) \in Q$, $V(i, j) = v$ for the unique $v \in V_1(i)$, and
- For all $(i, j) \in Q$, M is defined as follows:

For all $i \in \{1, \ldots, k_1\}$, let the distribution $M_{(i,\bot)}$ (a vector describing probabilities of going from $(i, \bot)$ to all the other states of Q) be defined as a distribution ϱ such that there exists $\pi \in [0, 1]^{k_1}$, $\varphi_1(i)(\pi)$, and for all $i' \in \{1, \ldots, k_1\}$, $\varrho((i', \bot)) = \pi(i')$, and for all $j' \in \{1, \ldots, k_2\}$, $j' \neq \bot$, $\varrho((i', j')) = 0$.

For all (i, j) in case 1 above, the vector $M_{(i,j)}$ is defined as a distribution ϱ such that there exists $\pi \in [0, 1]^{k_1}$, $\varphi_1(i)(\pi)$, and for all $i' \in \{1, \ldots, k_1\}$, $\varrho((i', \bot)) = \pi(i')$, and for all $j' \in \{1, \ldots, k_2\}$, $j' \neq \bot$, $\varrho((i', j')) = 0$.

For all (i, j) in case 2 above, there exists a distribution $\pi \in [0, 1]^{k_1}$ that can not be redistributed. $M_{(i,j)}$ is then defined as the distribution ϱ such that for all $i' \in \{1, \ldots, k_1\}$ and $j' \in (\{1, \ldots, k_2\} \cup \{\bot\})$,

$$\varrho((i', j')) = \begin{cases} \pi(i') & \text{if } \mathsf{succ}_{(i,j)}(i') = j' \text{ (possibly } \bot) \\ 0 & \text{otherwise.} \end{cases}$$

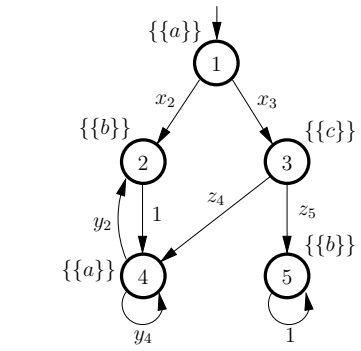
For all (i, j) in case 3 above, $M_{(i,j)}$ is defined as a distribution ϱ such that there exists $\pi \in [0, 1]^{k_1}$ such that $\varphi_1(i)(\pi)$, and for all $i' \in \{1, \ldots, k_1\}$ and $j' \in (\{1, \ldots, k_2\} \cup \{\bot\})$,

$$\varrho((i', j')) = \begin{cases} \pi(i') & \text{if } \mathsf{succ}_{(i,j)}(i') = j' \ (j' \text{ could be } \bot) \\ 0 & \text{otherwise.} \end{cases}$$

Example 9. *Consider CMCs S_1 and S_2 given in Fig. 13a and Fig. 13b, respectively. These CMCs are consistent, in single valuation normal form, and deterministic. Moreover, $S_1 \not\preceq S_2$ with $\mathcal{R} = \{(2, 2'), (4, 4'), (5, 5')\}$ being the relation after termination of the fixpoint loop of the weak refinement algorithm. The above witness generation algorithm computes the MC P in Fig. 13c.*

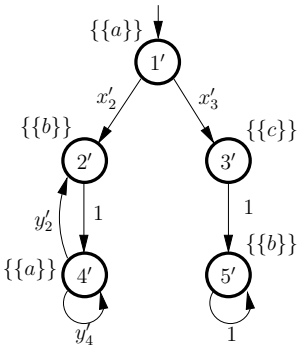
We now show that the above procedure indeed computes a witness for the absence of weak refinement.

Theorem 11. *Let $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be consistent, deterministic CMCs in single valuation normal form, such that $S_1 \not\preceq S_2$ and $A_2 \subseteq A_1$. For the MC $P = \langle \{1, \ldots, k_1\} \times (\{1, \ldots, k_2\} \cup \{\bot\}), o, M, A, V \rangle$ generated by the above algorithm, it holds that $P \in [\![S_1]\!]$ and $P \notin [\![S_2]\!]$.*



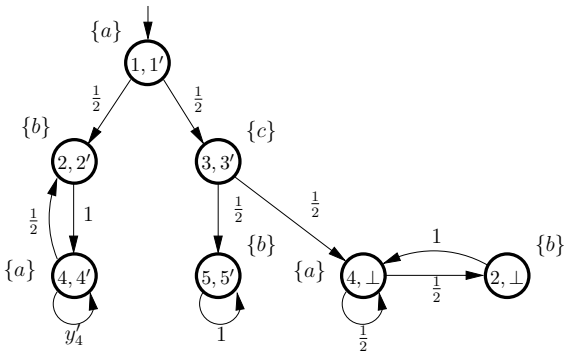
$\varphi(1)(x) \equiv x_2 \geq \frac{1}{2} \wedge x_3 \leq \frac{1}{2} \wedge x_2 + x_3 = 1$
$\varphi(4)(y) \equiv y_2 \geq \frac{1}{2} \wedge y_4 \leq \frac{1}{2} \wedge y_2 + y_4 = 1$
$\varphi(3)(z) \equiv z_4 = \frac{1}{2} \wedge z_5 = \frac{1}{2} \wedge z_4 + z_5 = 1$

(a) CMC S_1



$\varphi'(1')(x') \equiv x'_2 \geq \frac{1}{2} \wedge x'_3 \leq \frac{1}{2} \wedge x'_2 + x'_3 = 1$
$\varphi(4')(y') \equiv y'_2 \geq \frac{1}{2} \wedge y'_4 \leq \frac{1}{2} \wedge y'_2 + y'_4 = 1$

(b) CMC S_2



(c) MC P

Figure 13: Counterexample generation for refinement checking

Proof. We prove the two claims separately.

$P \in [\![S_1]\!]$: This is evident using the satisfaction relation,

$$\mathcal{R} = \{((i,j),k) \subseteq (\{1,\dots,k_1\} \times (\{1,\dots,k_2\} \cup \{\bot\})) \times \{1,\dots,k_1\} \mid i = k\}.$$

It is clear that $(o, o_1) \in \mathcal{R}$, since $o = (o_1, o_2)$.

$P \notin [\![S_2]\!]$:

We prove by contradiction that $P \not\models S_2$. Suppose that there exists a satisfaction relation $\mathcal{R}_S \subseteq (\{1,\dots,k_1\} \times (\{1,\dots,k_2\} \cup \{\bot\})) \times \{1,\dots,k_2\}$ such that $P \models S_2$. Let $\mathcal{R} \subseteq \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ be the relation such that $i \,\mathcal{R}\, j$ iff. $(i,j)\,\mathcal{R}_S\, j$.

By definition, we have $(o_1, o_2)\,\mathcal{R}_S\, o_2$, thus $o_1 \,\mathcal{R}\, o_2$. By hypothesis, we know that $S_1 \not\preceq S_2$, so $\mathcal{R}$ cannot be a refinement relation between S_1 and S_2. Since $o_1 \,\mathcal{R}\, o_2$, there must exist $i \in \{1,\dots,k_1\}$ and $j \in \{1,\dots,k_2\}$ such that $i \,\mathcal{R}\, j$ and the conditions for a refinement relation between i and j are broken.

Recall that $\mathcal{R}$ is a refinement relation if $i \,\mathcal{R}\, j$ implies:

1. $V_1(i)\downarrow_{A_2} \subseteq V_2(j)$ and
2. for any distribution $\pi \in [0,1]^{k_1}$ satisfying $\varphi_1(i)(\pi)$, there exists a correspondence matrix $\Delta \in [0,1]^{k_1 \times k_2}$ such that
   - for all S_1 states $1 \le i' \le k_1$, $\pi_{i'} \neq 0 \implies \sum_{j'=1}^{k_2} \Delta_{i'j'} = 1$;
   - $\varphi_2(j)(\pi\Delta)$ holds and
   - $\Delta_{i'j'} \neq 0 \Rightarrow i' \,\mathcal{R}\, j'$.

By definition of P, we have that $V((i,j)) \in V_1(i)$ and $|V_1(i)| = 1$. By $\mathcal{R}_S$, we have that $V((i,j))\downarrow_{A_2} \in V_2(j)$. Thus, since S_2 is in single valuation normal form, we have $V_1(i)\downarrow_{A_2} = V_2(j)$.

As a consequence, the second condition must not hold. Thus, there must exist a vector $\pi \in [0,1]^{k_1}$ such that $\varphi_1(i)(\pi) = 1$ and for all correspondence matrices $\Delta \in [0,1]^{k_1 \times k_2}$, at least one of the following conditions is broken:

(a) for all S_1 states $1 \le i' \le k_1$, $\pi_{i'} \neq 0 \implies \sum_{j'=1}^{k_2} \Delta_{i'j'} = 1$;

(b) $\varphi_2(j)(\pi\Delta)$ holds and

(c) $\Delta_{i'j'} \neq 0 \Rightarrow i' \,\mathcal{R}\, j'$.

We show that there exists a correspondence matrix Δ such that all three conditions hold, which leads to a contradiction and concludes the proof.

Since there must exist such a π we have by construction of P that $M_{(i,j)} = \varrho$ such that

$$\varrho((i',j')) = \begin{cases} \pi(i') & \text{if } \mathsf{succ}_{(i,j)}(i') = j' \\ 0 & \text{otherwise,} \end{cases}$$

and $\varrho((i',\bot)) = \pi(i')$ if $\mathsf{succ}_{(i,j)}(i') = \bot$.

Moreover, by $\mathcal{R}_S$, there exists a correspondence matrix Δ^S such that $\varphi_2(j)(\varrho\Delta^S) = 1$ and $\Delta^S_{(i',j'),l'} > 0 \Rightarrow (i',j')\,\mathcal{R}_S\,l'$.

By definition of the function $\mathsf{succ}_{(i,j)}$, if there exists $i' \in \{1,\dots,k_1\}$ such that $\varrho(i',\bot) > 0$, then there is no state $l' \in \{1,\dots,k_2\}$ such that $[\varrho\Delta^S]_{l'} > 0$ and $V((i',\bot))\downarrow_{A_2} \in V_2(l')$. This breaks the definition of a satisfaction relation. Thus for all i', we have that $\varrho(i',\bot) = 0$.

Let $\Delta \in [0,1]^{k_1\times k_2}$ be the correspondence matrix such that $\Delta_{i',j'} = \Delta^S_{(i',j'),j'}$ if $\mathsf{succ}_{(i,j)}(i') = j'$ and 0 otherwise. Since $\varrho(i',\bot) = 0$ for all i', we have that $\sum_{1\le l'\le k_2}\Delta_{i',l'}$
$= \sum_{1\le l'\le k_2}\Delta^S_{(i',j'),l'}$ ($\Delta^S_{(i',j'),l'}$ must be 0 whenever $l' \neq \mathsf{succ}_{(i,j)}(i')$). As a consequence, whenever $\pi(i') > 0$, we have $\sum_{1\le l'\le k_2}\Delta_{i',l'} = 1$ because of $\mathcal{R}_S$. Thus condition (a) holds.

Moreover, by construction, we have that for all $l \in \{1,\dots,k_2\}$,

$$\begin{aligned}
[\pi\Delta]_l &= \sum_{1\le i'\le k_1} \pi(i')\Delta_{i',l} \\
&= \sum_{i' \mid \mathsf{succ}_{(i,j)}(i')=l} \varrho(i',l)\Delta^S_{(i',l),l} \\
&= \sum_{(i',j')\in\{1,\dots,k_1\}\times(\{1,\dots,k_2\}\cup\{\bot\})} \varrho(i',j')\Delta^S_{(i',j'),l} \\
&= [\varrho\Delta^S]_l
\end{aligned}$$

Thus $\pi\Delta = \varrho\Delta^S$ and $\varphi_2(j)(\pi\Delta) = 1$. As a consequence, condition (b) holds.

Finally, let $i' \in \{1,\dots,k_1\}$ and $j' \in \{1,\dots,k_2\}$ be states such that $\Delta_{i',j'} > 0$. By construction, $\Delta_{i',j'} = \Delta^S_{(i',j'),j'}$, thus $\Delta^S_{(i',j'),j'} > 0$, and we obtain by $\mathcal{R}_S$ that $(i',j')\,\mathcal{R}_S\,j'$. As a consequence, $i'\,\mathcal{R}\,j'$ and condition (c) holds.

□

Checking Determinism First we construct values $v_{i'j'}$ for $(i',j') \in \{1,\dots,k_1\}\times\{1,\dots,k_2\}$ and $i' \neq j'$, such that

$$\forall i',j' : i' \neq j' \wedge V_1(i') \cap V_1(j') \neq \emptyset \Rightarrow v_{i'j'} = 0 \wedge \tag{7}$$
$$\forall i',j' : i' \neq j' \wedge V_1(i') \cap V_1(j') = \emptyset \Rightarrow v_{i'j'} = 1. \tag{8}$$

The equation passed to Z3 is given hereafter:

$$\begin{aligned}
&\forall x,y\in[0,1]^{k_1} : \varphi_1(i)(x) \wedge \varphi_1(i)(y) \Rightarrow && (9)\\
&\qquad \forall (i',j') \in \{1,\dots,k_1\}\times\{1,\dots,k_2\}, i' \neq j' \exists v_{i'j'} : && (10)\\
&\qquad \text{the values satisfy Eq. (7)–(8)} \wedge \\
&\qquad \forall i',j' : \left[i' \neq j' \wedge x_{i'} > 0 \wedge y_{j'} > 0\right] \Rightarrow v_{i'j'} = 1.
\end{aligned}$$

Computing The Constraint Abstraction We first remark that the state abstraction can be implemented by directly following definition, without relying on Z3. We thus do not discuss this part of the implementation. Rather, we focus on constraint abstraction that we introduced in Section 5.2. Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a deterministic CMC in single valuation normal form. The tool is capable of computing the constraint abstraction, $\chi(S) = \langle \{1, \ldots, k\}, o, \varphi', A, V \rangle$ for S. In order to compute $\varphi'(i)$, the tool has to compute the lower l_j and upper u_j bounds of the interval labeling each transition to a successor state $j \in \{1, \ldots, k\}$. This is done by passing the following equation to Z3:

$$
\begin{aligned}
&\forall 1 \leq j \leq k : (0 \leq l_j \leq u_j \leq 1) \wedge \\
&\Big(\forall x \in [0,1]^k : \varphi(i)(x) \Rightarrow \forall 1 \leq j \leq k : l_j \leq x_j \leq u_j\Big) \wedge \\
&\forall (l'_1, u'_1, \ldots, l'_k, u'_k) \in [0,1]^k : \\
&\qquad \Big[\Big(\Big(\forall x \in [0,1]^k : \varphi(i)(x) \Rightarrow \forall 1 \leq j \leq k : l'_j \leq x_j \leq u'_j \Big) \wedge \\
&\qquad\qquad (\forall 1 \leq j \leq k : 0 \leq l_j \leq u_j \leq 1) \Big) \Rightarrow \\
&\qquad \big(\forall 1 \leq j \leq k : l'_j \leq l_j \wedge u_j \leq u'_j\big) \Big].
\end{aligned}
$$

Notice that the l_j's and u_j's are not quantified, and are thus free. Using the above equations, unique values for l_j and u_j can be computed. This is done by using the model generation functionality and quantifier elimination, both provided by Z3. The constraint $\varphi'(i)$ is thus defined to be

$$
\varphi'(i)(x) \equiv \bigwedge_{j=1}^{k} x_j \in [l_j, u_j] \wedge \sum_{j=1}^{k} x_j = 1. \tag{11}
$$

7 Experiments

We performed experiments on randomly generated CMCs. For generating those CMCs, we first define simple and elaborated constraints for any state i. Given i, we define the transitions to the successor states $i+1$ and $i+2$ as follows (transitions to successors that are not mentioned are unconstrained)

- simple:
  - $x_{i+1} \geq 7/10 \wedge x_{i+2} \leq 3/10$,
  - $x_{i+1} = 7/10 \wedge x_{i+2} = 3/10$, and
  - $x_{i+1} = 1.0$
- more elaborate:
  - $x_{i+1} \geq 3/10 \wedge x_{i+1} \leq 4/10$,

– true, and
– $x_{i+1} = 1.0 \vee (x_{i+1} \geq 7/10 \wedge x_{i+2} \leq 3/10)$

Given a number of states, and whether or not we are interested in simple or more elaborate constraints, we generate an atomic proposition alphabet A on 5–10 members each, state valuations consisting of up to 0–4 members of 2^A, and a random choice between constraint designs for each transition.

Tables 4, 5, 6, 7, and 8 illustrate the execution times for different operations. The tests are performed on an Intel Core 2 Duo 2.2 GHz with 4 GB RAM running Windows 7 x64, using version 2.18 of the Z3 API. Three execution times are reported, as the experiment was repeated three times, with different randomly generated instances. A question mark (?) means that the specific random input file did not stop executing within 5 minutes.

Regarding abstraction, we made a test of abstraction from a CMC with 500 states to CMCs with 5, 50, and 100 states, respectively. Given a number a of abstract states, we define the state abstraction function α_a as follows:

$$\alpha_a : \begin{cases} \left\{1, \ldots, \frac{500}{a}\right\} & \mapsto \quad 1 \\ \left\{\frac{500}{a} + 1, \ldots, \frac{500 \cdot 2}{a}\right\} & \mapsto \quad 2 \\ & \ldots \\ \left\{\frac{500(a-1)}{a} + 1, \ldots, \frac{500 \cdot a}{a}\right\} & \mapsto \quad a \end{cases}$$

As one can see in Table 6, computation time increases linearly with precision (i.e., with the number of states of the abstraction). Finally, Table 7 proposes some results for abstraction by IMCs.

8 Related Work and Concluding Remarks

In [63], we have presented CMCs—a new model for representing a possibly infinite family of MCs. Unlike the previous attempts [46, 48], our model is closed under many design operations, including composition and conjunction. We have studied these operations as well as several classical compositional reasoning properties, showing that, among others, the CMC specification theory is equipped with a complete refinement relation (for deterministic specifications), which naturally interacts with parallel composition, synchronization and conjunction. We have also demonstrated how our framework can be used to obtain properties for less expressive languages, by using reductions.

This paper proposes new results for CMCs: (1) the first proof that IMCs are not closed under conjunction, which establish CMCs as a leading behavioural specification for stochastic systems, (2) a series of abstraction techniques to ease the computation process, and (3) the first tool for CMCs. Our tool relies on an encoding of all the operations within the formalism of the Z3 solver.

Two recent contributions [48, 49] are related to our work. Fecher et al. [48] propose a model checking procedure for PCTL [98] and Interval Markov Chains (other procedures

recently appear in [77, 78]), which is based on weak refinement. However, our objective is not to use CMCs within a model checking procedure for probabilistic systems, but rather as a specification theory.

Recently Katoen and coauthors [49] have extended Fecher's work to *Interactive* Markov Chains, a model for performance evaluation [99, 100]. Their abstraction uses the continuous time version of IMCs [47] augmented with may and must transitions, very much in the spirit of [42].

Over the years process algebraic frameworks have been proposed for describing and analyzing probabilistic systems based on Markov Chains (MCs) and Markov Decision Processes [29, 68, 92]. Also a variety of probabilistic logics have been developed for expressing properties of such systems, e.g., PCTL [18]. Both traditions support refinement between specifications using various notions of probabilistic simulation [46, 48] and, respectively, logical entailment [101]. Whereas the process algebraic approach favors structural composition (parallel composition), the logical approach favors logical composition (conjunction). Neither of the two supports *both* structural and logical composition.

Future work As a future work, we would also like to define a quotient relation for CMCs, presumably building on results presented in [103]. The quotienting operation is of particular importance for component reuse. One could also investigate applicability of our approach in model checking procedures, in the same style as Fecher and coauthors have used IMCs for model checking PCTL [48] or by extending the stochastic version of Hennessy-Milner logic [16]. Finally, it would be interesting to extend our composition operation by considering products of dependent probability distributions in the spirit of [110]. Of course, before doing so, our first objective is to improve our implementation. Our very first step will be to consider other solvers such as the computer algebra system Maple [114] in order to perform composition automatically. We shall also implement heuristics to reduce computation time [115].

Table 3: Operators implemented in the tool

Code	Meaning
`S1 wref S2`	Decide if `S1` weakly refines `S2`
`show S1`	Print `S1` to the console
`D(S1)`	Decide if `S1` is deterministic
`beta*(S1)`	The pruned version of `S1`
`N(S1)`	The normalized version of `S1`
`S1 and S2`	The conjunction of `S1` and `S2`
`C(S1)`	Decide if `S1` is consistent
`alpha(S1,<set definition>)`	The state abstraction of `S1`
`chi(S1)`	The constraint abstraction of `S1`

Table 4: Weak refinement

CMC 1		CMC 2		
states	simple	states	simple	time
10	yes	10	yes	6/297/8718 ms
10	no	10	yes	6/1760/3897 ms
10	yes	10	no	40/135/14967 ms
10	no	10	no	1201/7459/? ms
15	yes	15	yes	59467/?/? ms

CMC		
states	simple	time
10	yes	42/51/56 ms
10	no	42/46/308 ms
15	yes	162/2360/2364 ms
15	no	317/2314/2348 ms

Table 5: Determinism

CMC			
states	abtract states	simple	time
500	5	yes	1981/2936/5533 ms
500	5	no	1924/2608/5900 ms
500	50	yes	11521/11556/11575 ms
500	50	no	11152/11281/11350 ms
500	100	yes	25192/26217/26625 ms
500	100	no	26031/26234/26444 ms

Table 6: State abstraction

CMC		
states	simple	time
3	yes	160/214/236 ms
3	no	184/192/193 ms
5	yes	414/423/979 ms
5	no	718/735/831 ms

Table 7: Constraint abstraction

CMC		
states	simple	time
10	yes	173/181/196 ms
10	no	31/46/87 ms
15	yes	187/511/1871 ms
15	no	83/110/119 ms

Table 8: Consistency

Paper D – Abstract Probabilistic Automata

Benoit Delahaye
INRIA/IRISA, Rennes

Joost-Pieter Katoen
RWTH Aachen University

Kim G. Larsen
Aalborg University

Axel Legay
INRIA/IRISA, Rennes

Mikkel L. Pedersen
Aalborg University

Falak Sher
RWTH Aachen University

Andrzej Wasowski
IT University, Copenhagen

1 Abstract

Probabilistic Automata (PAs) are a widely-recognized mathematical framework for the specification and analysis of systems with non-deterministic and stochastic behaviors. This paper proposes Abstract Probabilistic Automata (APAs), that is a novel abstraction model for PAs. In APAs uncertainty of the non-deterministic choices is modeled by may/must modalities on transitions while uncertainty of the stochastic behaviour is expressed by (underspecified) stochastic constraints. We have developed a complete abstraction theory for PAs, and also propose the first specification theory for them. Our theory supports both satisfaction and refinement operators, together with classical stepwise design operators. In addition, we study the link between specification theories and abstraction in avoiding the state-space explosion problem.

2 Introduction

Probabilistic Automata (PAs) constitute a mathematical framework for the specification and analysis of non-deterministic probabilistic systems. They have been developed by Segala [52] to model and analyze asynchronous, concurrent systems with discrete probabilistic choice in a formal and precise way. PAs are akin to Markov decision processes (MDPs). A detailed comparison with models such as MDPs, as well as generative and reactive probabilistic transition systems is given in [116]. PAs are recognized as an adequate formalism for randomized distributed algorithms and fault tolerant systems. They are used as semantic model for formalisms such as probabilistic process algebra [117] and a probabilistic variant of Harel's statecharts [118]. An input-output version of PAs is the basis of PIOA and variants thereof [119, 120]. PAs have been enriched with notions such as weak and strong (bi)simulations [52], decision algorithms for these notions [121] and a statistical testing theory [122]. This paper brings two new contributions to the field of probabilistic automata: the theories of *abstraction* and of *specification*.

Abstraction is pivotal to combating the state explosion problem in the modeling and verification of realistic systems such as randomized distributed algorithms. It aims at model reduction by collapsing sets of concrete states to abstract states, e.g., by partitioning the concrete state space. This paper presents a three-valued abstraction of PAs. The main design principle of our model, named *Abstract Probabilistic Automata* (APAs), is to abstract sets of distributions by constraint functions. This generalizes earlier work on interval-based abstraction of probabilistic systems [46, 47, 48]. To abstract from action transitions, we introduce *may* and *must* modalities in the spirit of modal transition systems [35]. If all states in a partition p have a must-transition on action a to some state in partition p', the abstraction yields a must-transition between p and p'. If some of the p-states have no such transition while others do, it gives rise to a may-transition between p and p'. Our model shall be viewed as a combination of both Modal Automata [42] and Constraint Markov Chains (CMC) [63] that are abstractions for transition systems and Markov Chains, respectively.

We also propose the first specification theory for PAs, equipped with all essential ingredients of a compositional design methodology: a satisfaction relation (to decide whether a PA is an implementation of an APA), a consistency check (to decide whether the specification admits an implementation), a refinement (to compare specifications in terms of inclusion of sets of implementations), logical composition (to compute the intersection of sets of implementations), and structural composition (to combine specifications). Our framework also supports incremental design [91]. To the best of our knowledge, the theory of APAs is the first specification theory for PAs in where both logical and structural compositions can be computed within the same framework.

Our notions of refinement and satisfaction are, as usual, characterized in terms of inclusion of sets of implementations. One of our main theorems shows that for the class of deterministic APAs, refinement coincides with inclusion of sets of implementations. This latter result is obtained by a reduction from APAs to CMCs, for which a similar

result holds. Hence, APAs can also be viewed as a specification theory for Markov Chains (MCs). The model is as expressive as CMCs, and hence more expressive than other theories for stochastic systems such as Interval Markov Chains [46, 48, 61].

Our last contribution is to propose an *abstraction-based* methodology that allows to simplify the behavior of APAs with respect to the refinement relation – such an operation is crucial to avoid state-space explosion. We show that our abstraction preserves weak refinement, and that weak refinement is a pre-congruence with respect to parallel composition. These results provide the key ingredients to allow *compositional* abstraction of PAs.

Organisation of the paper. In Section 3, we introduce the concepts of APAs and a satisfaction relation with respect to PAs. We also propose a methodology to decide whether an APA is consistent. Refinement relations and abstraction of APAs are discussed in Section 4. Other compositional reasoning operators such as conjunction and composition as well as their relation with abstraction are presented in Section 5. Section 6 discusses the relation between CMCs and APAs and proposes a class of deterministic APAs for which strong and weak refinements coincide with inclusion of sets of implementations. Finally, Section 7 concludes the paper and proposes directions for future research. Due to space limitation, proofs and larger examples are given in a long version of this paper [123].

3 Specifications and Implementations

We now introduce the main models of the paper: first *Probabilistic Automata*, and then the new abstraction—*Abstract Probabilistic Automata*.

Implementations. A PA [52] resembles a non-deterministic automaton, but its transitions target probability distributions over states instead of single states. Hence, PAs can be seen as a combination of Markov Chains and non-deterministic automata or as Markov Decision Processes allowing non-determinism.

Definition 1. (Probabilistic automata) *A* probabilistic automaton *is a tuple* (S, A, L, AP, V, s_0)*, where:*

- S *is a finite set of states with initial state* $s_0 \in S$*,*
- A *is a finite set of actions,*
- $L\colon S \times A \times Dist(S) \to \mathbb{B}_2$ *is a* two-*valued transition function,*
- AP *is a finite set of valuations, and*
- $V\colon S \to 2^{AP}$ *is a state-labeling function.*

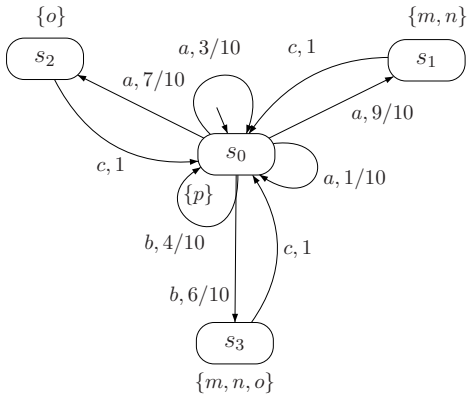


Figure 1: An example PA

Here $\mathbb{B}_2 = \{\bot, \top\}$, with $\bot < \top$. $L(s, a, \mu)$ identifies the *transition* of the automaton: $\top$ indicates its presence and $\bot$ indicates its absence. We write $s \xrightarrow{a} \mu$ meaning $L(s, a, \mu) = \top$. In the rest of the paper, we assume that PAs are *finitely branching*, i.e., for any state s, the number of pairs (a, μ) such that $s \xrightarrow{a} \mu$ is finite. The *labeling function* V indicates the propositions (or properties) that are valid in a state. A *Markov Chain* (MC) is a PA, where, for each $s \in S$, there exists exactly one triple (s, a, μ) such that $L(s, a, \mu) = \top$.

Example 1. *Figure 1 presents a PA with* $L(s_0, a, \mu) = \top$*, where* $\mu(s_0) = 3/10$ *and* $\mu(s_2) = 7/10$*. We adopt a notational convention that represents* $L(s_0, a, \mu) = \top$ *by a set of arrows with tails located close to each other on the boundary of* s_0*, and heads targeting the states in the support of* μ*.*

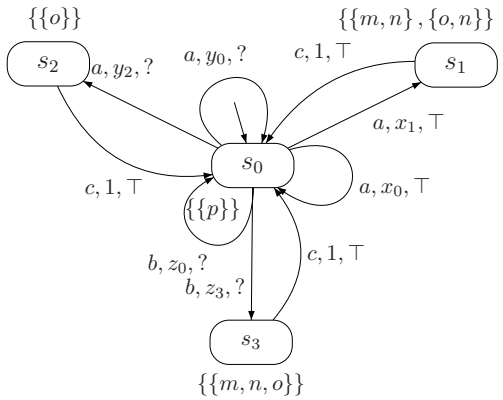
In state s_0*, a non-deterministic choice takes places on action a between the distributions* μ *and* μ' *with* $\mu'(s_0) = 1/10$ *and* $\mu'(s_1) = 9/10$*.*

Specifications. A Constraint Markov Chain (CMC) [63] is a MC equipped with a constraint on the next-state probabilities from any state. Roughly speaking, an implementation for a CMC is thus a MC, whose next-state probability distribution satisfies the constraint associated with each state. Let $Sat(\varphi)$ denote the set of distributions that satisfy constraint function φ, and $C(S)$ the set of constraint functions defined on state space S.

A *Modal Automaton* [35, 124] is an automaton whose transitions are typed with *may* and *must* modalities. Informally, a must transition is available in every model of the specification, while a may transition needs not be.

An *Abstract Probabilistic Automaton* (APA) is an abstraction that represents a possibly infinite set of PAs. APAs combine Modal Automata and CMCs – the abstractions for labelled transition systems and Markov Chains, respectively.

Definition 2. *An* abstract PA *is a tuple* (S, A, L, AP, V, s_0) *such that:*



$$\varphi_x \equiv x_1 \geq 0.9 \wedge x_0 + x_1 = 1$$
$$\varphi_y \equiv y_2 \leq 0.8 \wedge y_0 + y_2 = 1$$
$$\varphi_z \equiv z_3 \geq 0.5 \wedge z_0 + z_3 = 1$$

Figure 2: An example APA

- *S, A, AP and s_0 are defined as before*
- *$L : S \times A \times C(S) \longrightarrow \mathbb{B}_3$ is a* three-*valued state-constraint function, and*
- *$V : S \longrightarrow 2^{2^{AP}}$ maps a state onto a set of admissible valuations.*

Here, $\mathbb{B}_3 = \{\bot, ?, \top\}$ denotes a *complete lattice* with the following ordering $\bot < ? < \top$ and meet ($\sqcap$) and join ($\sqcup$) operators. A CMC is thus an APA, where for each $s \in S$, there exists exactly one triple (s, a, φ) such that $L(s, a, \mu) = \top$, while an *Interval Markov Chain* (IMC) [46] is a CMC whose constraints are disjunctions of intervals. The labeling $L(s, a, \varphi)$ identifies the "type" of the constraint function $\varphi \in C(S)$: $\top$, ? and $\bot$ indicate a *must*, a *may* and the absence of a constraint function, respectively. We could have limited ourselves to constraints denoting unions of intervals of probability values. However, as we shall soon see, polynomial constraints are needed to support *both* conjunction *and* parallel composition. Like for CMCs, states of an APA are labeled with a set of subsets of atomic propositions. A single set of propositions represents properties that should be satisfied by an implementation state. A powerset models a disjunctive choice of properties. Later, we shall see that any APA whose states are labelled with a set of subsets of atomic propositions can be turned into an equivalent (in the sense of implementations set) APA whose states are labeled with a set that contains only a single subset of AP.

Finally, observe that a PA is an APA in which every transition (s, a, μ) is represented by a *must*-transition (s, a, φ) with $Sat(\varphi) = \{\mu\}$, and each state-label consists of a single set of propositions.

Example 2. *Consider the APA N given in Figure 2. State s_0 has three outgoing transitions: a* must *a-transition (s_0, a, φ_x), a* may *a-transition (s_0, a, φ_y), and a* may

b-transition (s_0, b, φ_z)*. Due to the constraint, each of these transitions can cover several transitions in a concrete implementation PA. As an example, the a-transition* $(s_0, a, (1/10, 9/10, 0, 0))$ *of the PA given in Figure 1 is satisfying the* must *a-transition* (s_0, a, φ_x)*.*

In the rest of the paper we distinguish deterministic APAs. The distinction will be of particular importance when comparing APAs in Section 4.1. In APAs, the nondeterminism can arise due to sets of valuations in states, or due to actions that label transitions:

Definition 3 (Deterministic APA)**.** *An APA* $N = (S, A, L, AP, V, s_0)$ *is*

- *action-deterministic, if* $\forall s \in S. \forall a \in A.\ |\{\varphi \in C(S) \mid L(s, a, \varphi) \neq \bot\}| \leq 1$.
- *valuation-deterministic, if* $\forall s \in S. \forall a \in A. \forall \varphi \in C(S)$ *with* $L(s, a, \varphi) \neq \bot$*:*

$$\forall \mu', \mu'' \in Sat(\varphi), s', s'' \in S,\ \left(\mu'(s') > 0 \wedge \mu''(s'') > 0\ \Rightarrow V(s') \cap V(s'') = \emptyset\right).$$

N is deterministic iff it is both action-deterministic and valuation-deterministic.

Satisfaction. We relate APA specifications to PAs implementing them, by extending the definitions of satisfaction introduced in [46]. We start with the following definition that relates distributions between set of states. We use $Dist(S)$ to denote a set of *probability distributions* on the finite set S in the usual way.

Definition 4. $(\Subset_R^\delta)$ *Let* S *and* S' *be non-empty sets of states. Given* $\mu \in Dist(S)$*,* $\mu' \in Dist(S')$*, a function* $\delta : S \to (S' \to [0, 1])$*, and a binary relation* $R \subseteq S \times S'$*,* μ *is simulated by* μ' *with respect to* R *and* δ*, denoted as* $\mu \Subset_R^\delta \mu'$*, iff*

1. *for all* $s \in S$*, if* $\mu(s) > 0$*, then* $\delta(s)$ *is a distribution on* S'*,*
2. *for all* $s' \in S'$*,* $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$*, and*
3. *if* $\delta(s)(s') > 0$*, then* $(s, s') \in R$*.*

In the rest of the paper, we write $\mu \Subset_R \mu'$ *iff there exists a function* δ *such that* $\mu \Subset_R^\delta \mu'$*. Such* δ *is called a* correspondence function*.*

We are now ready to define the satisfaction relation between PAs and APAs.

Definition 5. (Satisfaction relation) *Let* $P = (S, A, L, AP, V, s_0)$ *be a PA and* $N = (S', A, L', AP, V', s_0')$ *be an APA.* $R \subseteq S \times S'$ *is a* satisfaction relation *iff, for any* $(s, s') \in R$*, the following conditions hold:*

1. $\forall a \in A, \forall \varphi' \in C(S') : L'(s', a, \varphi') = \top \Longrightarrow \exists \mu \in Dist(S) : L(s, a, \mu) = \top$ *and* $\exists \mu' \in Sat(\varphi')$ *such that* $\mu \Subset_R \mu'$*,*

2. $\forall a \in A, \forall \mu \in Dist(S) : L(s,a,\mu) = \top \implies \exists \varphi' \in C(S') : L'(s',a,\varphi') \neq \bot$ *and* $\exists \mu' \in Sat(\varphi')$ *such that* $\mu \Subset_R \mu'$, *and*

3. $V(s) \in V'(s')$.

We say that P satisfies N, denoted $P \models N$, iff there exists a satisfaction relation relating s_0 and s'_0. If $P \models N$, P is called an implementation *of N.*

Thus, a PA P is an implementation of an APA N iff any must-transition of N is matched by a must-transition of P that agrees on the probability distributions specified by the constraint, and reversely, P does not contain must-transitions that do not have a corresponding (may- or must-) transition in N. The set of all implementations of N is given by $[\![N]\!] = \{P \mid P \models N\}$.

Example 3. *The relation $R = \{(s_0,s_0),(s_1,s_1),(s_2,s_2),(s_3,s_3)\}$ is a satisfaction relation between the PA P given in Figure 1 and the APA N of Figure 2.*

Consistency. An APA N is *consistent* iff it admits at least one implementation. We say that a state s is *consistent* if $V(s) \neq \emptyset$ and $L(s,a,\varphi) = \top \implies Sat(\varphi) \neq \emptyset$. An APA is *locally consistent* if all its states are consistent. It is easy to see that a locally consistent APA is also consistent, i.e. has at least one implementation. However, inconsistency of a state does not imply inconsistency of the specification. In order to decide whether a specification is consistent, we proceed as usual and propagate inconsistent states with the help of a *pruning operator* β that filters out distributions leading to inconsistent states. This operator is applied until a fixed point is reached, i.e., until the specification does not contain inconsistent states (it is locally consistent). See [123] for details.

Theorem 1. *For any APA N, it holds: $[\![N]\!] = [\![\beta(N)]\!]$.*

As the set of states of N is finite, the fixed point computation will always terminate. By the above theorem, we have that $[\![N]\!] = [\![\beta^*(N)]\!]$.

4 Abstraction and Refinement

In this section we introduce *Refinement* that allows to compare APAs. We also propose an *abstraction-based* methodology that permits to simplify the behavior of APAs with respect to the refinement relation.

4.1 Refinement

A refinement compares APAs with respect to their sets of implementations. More precisely, if APA N refines APA N', then the set of implementations of N should be included in the one of N'. The ultimate refinement relation that can be defined between APAs is thus *Thorough Refinement* that exactly corresponds to inclusion of sets of implementations.

Definition 6. (Thorough refinement) *Let* $N = (S, A, L, AP, V, s_0)$ *and* $N' = (S', A, L', AP, V', s'_0)$ *be APAs. We say that* N *thoroughly refines* N'*, denoted* $N \preceq_T N'$*, iff* $[\![N]\!] \subseteq [\![N']\!]$.

For most specification theories, it is known that deciding thorough refinement is computationally intensive (see for example [43]). For many models such as Modal automata or CMCs, one can partially avoid the problem by working with a syntactical notion of refinement. This definition, which is typically strictly stronger than thorough refinement, is easier to check. The difference between syntactic and semantic refinements resembles the difference between simulations and trace inclusion for transition systems.

We consider two syntactical refinements. These relations extend two well known refinement relations for CMCs and IMCs by combining them with the refinement defined on modal automata. We start with the strong refinement.

Definition 7. (Strong refinement) *Let* $N = (S, A, L, AP, V, s_0)$ *and* $N' = (S', A, L', AP, V', s'_0)$ *be APAs.* $R \subseteq S \times S'$ *is a* strong refinement *relation iff, for all* $(s, s') \in R$*, the following conditions hold:*

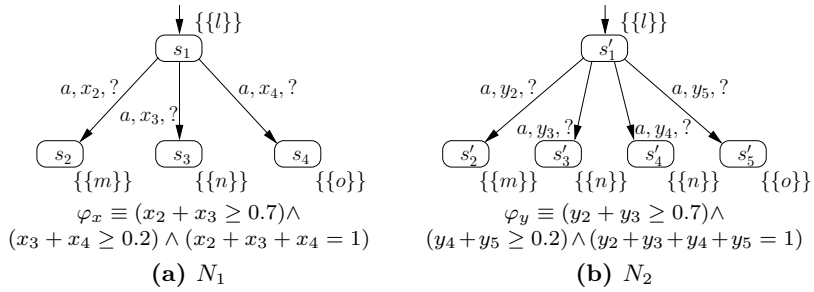
1. $\forall a \in A. \forall \varphi' \in C(S'). L'(s', a, \varphi') = \top \implies \exists \varphi \in C(S). L(s, a, \varphi) = \top$ *and there exists a correspondence function* $\delta : S \to (S' \to [0, 1])$ *such that* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset^{\delta}_{R} \mu'$,
2. $\forall a \in A. \forall \varphi \in C(S). L(s, a, \varphi) \neq \bot \implies \exists \varphi' \in C(S'). L'(s', a, \varphi') \neq \bot$ *and there exists a correspondence function* $\delta : S \to (S' \to [0, 1])$ *such that* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset^{\delta}_{R} \mu'$*, and*
3. $V(s) \subseteq V'(s')$.

We say that N *strongly refines* N'*, denoted* $N \preceq_S N'$*, iff there exists a strong refinement relation relating* s_0 *and* s'_0.

Observe that strong refinement imposes a "fixed-in-advance" δ in the simulation relation between distributions. This assumption is lifted with the definition of *weak refinement*:

Definition 8. (Weak refinement) *Let* $N = (S, A, L, AP, V, s_0)$ *and* $N' = (S', A, L', AP, V', s'_0)$ *be APAs.* $R \subseteq S \times S'$ *is a* weak refinement *relation iff, for all* $(s, s') \in R$*, the following conditions hold:*

1. $\forall a \in A. \forall \varphi' \in C(S'). L'(s', a, \varphi') = \top \implies \exists \varphi \in C(S). L(s, a, \varphi) = \top$ *and* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset_{R} \mu'$,
2. $\forall a \in A. \forall \varphi \in C(S). L(s, a, \varphi) \neq \bot \implies \exists \varphi' \in C(S'). L'(s', a, \varphi') \neq \bot$ *and* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset_{R} \mu'$*, and*
3. $V(s) \subseteq V'(s')$.



(a) N_1 **(b)** N_2

Figure 3: APAs N_1 and N_2 such that $N_1 \preceq N_2$, but not $N_1 \preceq_S N_2$.

We say that N weakly refines N', denoted $N \preceq N'$, iff there exists a weak refinement relation relating s_0 and s'_0.

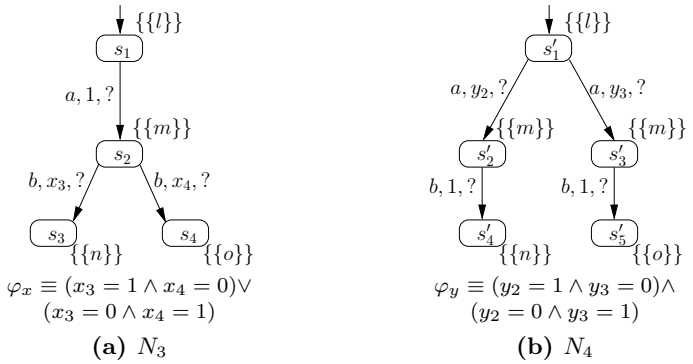
It is easy to see that the above definitions are combinations of the definitions of strong and weak refinement of CMCs with the *modal refinement* of Modal Automata. Hence algorithms for checking weak and strong refinements for APAs can be obtained by combining existing fixed-point algorithms for CMCs [62] and Modal Automata [42]. For the class of polynomial-constraint APAs, the upper bound for deciding weak/strong refinement is thus exponential in the number of states and doubly-exponential in the size of the constraints [62]. Both strong and weak refinement imply inclusion of sets of implementations. However, the converse is not true. The following theorem classifies the refinement relations.

Theorem 2. *Thorough refinement is strictly finer than weak refinement, and weak refinement is strictly finer than strong refinement.*

Proof. We present a sketch of the proof and refer to [123] for details. By definition, we have that $\preceq_S$ implies $\preceq$. By observing the definition of satisfaction relation, one can easily deduce that $\preceq_S$ and $\preceq$ imply $\preceq_T$. Consider now the APAs N_1 and N_2 given in Figure 3. It is easy to see that $N_1 \preceq N_2$. However, we have that $N_1 \not\preceq_S N_2$. Informally, one can see that State s'_3 and State s'_4 of N_2 both correspond to State s_3 of N_1. Thus, the probability mass x_3 of going to state s_3 in N_1 has to be distributed on s'_3 and s'_4 in order to match probabilities y_3 and y_4. The latter shall be achieved with the correspondence function δ that defines the refinement relation. The crucial point is that this correspondence function will depend on the exact value of x_3, thus δ cannot be precomputed and we have that $\preceq$, but not $\preceq_S$ holds.

Similarly, $\preceq$ does not imply $\preceq_T$. Consider the APAs N_3 and N_4 given in Figure 4. It is easy to see that $\preceq_T$ holds between N_3 and N_4. However, State s_2 of N_3 cannot refine State s'_2 or s'_3. Indeed, State s_2 has more implementations than s'_2 and s'_3 taken separately. □

We have just seen that thorough refinement is strictly finer than strong and weak refinement. In Section 6, we will propose a class of deterministic APAs on which the three relations coincide.



(a) N_3 **(b)** N_4

Figure 4: APAs N_3 and N_4

4.2 Abstraction

This section covers the abstraction of APA. The rationale is to partition the state space, i.e., group (disjoint) sets of states by a single abstract state. Let N and M be APA with state space S and S', respectively. An *abstraction* function $\alpha : S \to S'$ is a surjection. The inverse of abstraction function α is the *concretization* function $\gamma : S' \to 2^S$. The state $\alpha(s)$ denotes the abstract counterpart of state s while $\gamma(s')$ represents the set of all (concrete) states that are represented by the abstract state s'. Abstraction is lifted to distributions as follows. The abstraction of $\mu \in Dist(S)$, denoted $\alpha(\mu) \in Dist(S')$, is uniquely defined by $\alpha(\mu)(s') = \mu(\gamma(s'))$ for all $s' \in S'$.

Abstraction is lifted to sets of states, or sets of distributions in a pointwise manner. It follows that $\varphi' = \alpha(\varphi)$ iff $Sat(\varphi') = \alpha(Sat(\varphi))$. The abstraction of the product of constraint functions φ and φ' is given as $\alpha(\varphi \cdot \varphi') = \alpha(\varphi) \cdot \alpha(\varphi')$. These ingredients provide the basis to define the abstraction of an APA.

Definition 9. (Abstraction) *Given APA $N = (S, A, L, AP, V, s_0)$, the abstraction function $\alpha : S \to S'$ induces the APA $\alpha(N) = (S', A, L', AP, V', \alpha(s_0))$, where for all $a \in A$, $s' \in S'$ and $\varphi' \in C(S')$:*

$$L'(s', a, \varphi') = \begin{cases} \top & \text{if } \forall s \in \gamma(s') : \exists \varphi \in C(S) : L(s, a, \varphi) = \top, \text{ and} \\ & Sat(\varphi') = \alpha(\bigcup_{(s,\varphi) \in \gamma(s') \times C(S) : L(s,a,\varphi) = \top} Sat(\varphi)) \quad (a) \\ ? & \text{if } \exists s \in \gamma(s') : \exists \varphi \in C(S) : L(s, a, \varphi) \neq \bot, \text{ and} \\ & Sat(\varphi') = \alpha(\bigcup_{(s,\varphi) \in \gamma(s') \times C(S) : L(s,a,\varphi) \neq \bot} Sat(\varphi)) \quad (b) \\ \bot & \text{otherwise} \quad (c) \end{cases}$$

$$\text{and } V'(s') = \bigcup\nolimits_{\forall s \in \gamma(s')} V(s)$$

Item (a) asserts that if there are must transitions (s, a, φ) from all states $s \in \gamma(s')$, then the must transition (s', a, φ') represents the total behavior. Item (b) asserts that a may a-transition emanating from s' represents the total behaviour of all transitions

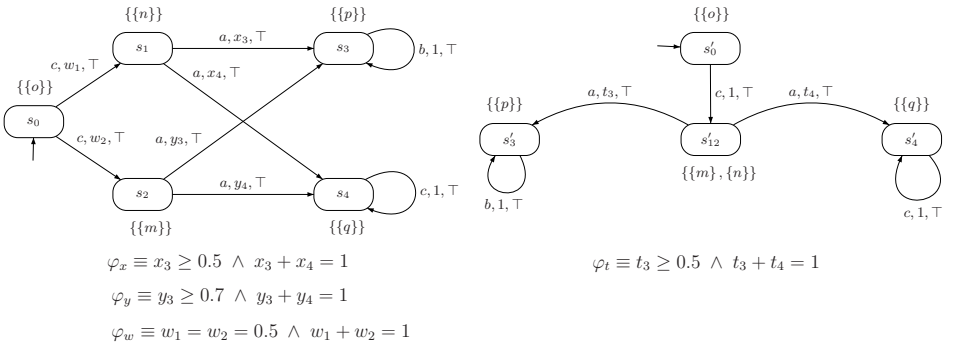


Figure 5: The APA N (left) is abstracted by the APA N' (right), i.e. $N' = \alpha(N)$

(s, a, φ) for $s \in \gamma(s')$, if not all states in $\gamma(s')$ have a must a-transition, and there is a a-transition on modality different from $\bot$. Item (c) asserts that if no state in $\gamma(s')$ has an a-transition, then s' also does not have an a-transition.

The result of abstracting APA N is the APA $\alpha(N)$ that is able to mimic all behaviours of N, but possibly exhibits more behaviour.

Lemma 3. *For any APA N, $\alpha(N)$ is an APA.*

Example 4. *Consider the APA $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP, V', s'_0)$ depicted in Fig. 5. Let the abstraction function $\alpha : S \to S'$ be given by $\alpha(s_0) = s'_0$, $\alpha(s_1) = s'_{12} = \alpha(s_2)$, $\alpha(s_3) = s'_3$ and $\alpha(s_4) = s'_4$. Both states s_1 and s_2 in N have a must a-transition. These are abstracted in N' by a single must a-transition satisfied by distributions in the union of satisfaction sets of φ_x and φ_y.*

Observe that the abstract version of an APA is always weaker in term of refinement than the original APA.

Theorem 4. *For any APA N and abstraction function α, $N \preceq \alpha(N)$.*

5 Compositional Reasoning

APAs can serve as a specification theory for systems with both non-deterministic and stochastic behaviors. Any good specification theory shall be equiped with a *conjunction operation* that allows to combine multiple requirements into a single specification, and a *composition operation* that allows specifications to be combined structurally. Studying these two operations for APAs is the subject of this section.

5.1 Conjunction

Conjunction, also called *logical composition*, allows combining two specifications into a single specification, that has the conjunctive behavior of the two operands. More

precisely, conjunction allows to compute the intersection of sets of implementations. In this paper, conjunction will be defined for action-deterministic APAs with the same action alphabet. The generalization to non-deterministic APAs with dissimilar alphabets, which is already known to be complex for the case of Modal Automata [82], is postponed for future work. The conjunction operation is a mix between the corresponding operation for modal automata and CMCs.

Definition 10 (Conjunction)**.** *Let* $N = (S, A, L, AP, V, s_0)$ *and* $N' = (S', A, L', AP, V', s'_0)$ *be action-deterministic APAs sharing actions sets and atomic propositions sets. The conjunction of* N *and* N' *is the APA* $N \wedge N' = (S \times S', A, \tilde{L}, AP, \tilde{V}, (s_0, s'_0))$ *such that*

- $\tilde{L}$ *is defined as follows. For all* $a \in A$ *and* $(s, s') \in S \times S'$*,*
  - *If there exists* $\varphi \in C(S)$ *such that* $L(s, a, \varphi) = \top$ *and for all* $\varphi' \in C(S')$*, we have* $L'(s', a, \varphi') = \bot$*, or if there exists* $\varphi' \in C(S')$ *such that* $L'(s', a, \varphi') = \top$ *and for all* $\varphi \in C(S)$*, we have* $L(s, a, \varphi) = \bot$*, then* $\tilde{L}((s, s'), a, \mathit{false}) = \top$*.*
  - *Else, if either for all* $\varphi \in C(S)$*, we have* $L(s, a, \varphi) = \bot$ *or for all* $\varphi' \in C(S')$*, we have* $L'(s', a, \varphi') = \bot$*, then for all* $\tilde{\varphi} \in C(S \times S')$*,* $\tilde{L}((s, s'), a, \tilde{\varphi}) = \bot$*.*
  - *Otherwise, for all* $\varphi \in C(S)$ *and* $\varphi' \in C(S')$ *such that* $L(s, a, \varphi) \neq \bot$ *and* $L'(s', a, \varphi') \neq \bot$*, define* $\tilde{L}((s, s'), a, \tilde{\varphi}) = L(s, a, \varphi) \sqcup L'(s', a, \varphi')$ *with* $\tilde{\varphi}$ *the new constraint in* $C(S \times S')$ *such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff*
    - *the distribution* $\mu : t \to \sum_{t' \in S'} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi)$*, and*
    - *the distribution* $\mu' : t' \to \sum_{t \in S} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi')$*.*
  - *Finally, for all other* $\tilde{\varphi}' \in C(S \times S')$*, let* $\tilde{L}((s, s'), a, \tilde{\varphi}') = \bot$*.*
- $\tilde{V}((s, s')) = V(s) \cap V'(s')$*.*

Observe that the conjunction of two action-deterministic APAs is an action-deterministic APA. The conjunction operation may introduce inconsistent states. Hence, any conjunction operation has to be followed by a pruning operation. Finally, observe that the conjunction of two APAs with interval constraints is not necessarily an APA with interval constraints, but could be an APA whose constraints are systems of linear inequalities (see [123] for an example).

The following theorem states that the pruned conjunction of two action-deterministic APAs matches their greatest lower bound with respect to refinement.

Theorem 5. *Let* N*,* N'*, and* N'' *be action-deterministic consistent APAs. It holds that* $\beta^*(N \wedge N') \preceq N$ *and, if* $N'' \preceq N$ *and* $N'' \preceq N'$*, then* $N'' \preceq \beta^*(N \wedge N')$*.*

5.2 Parallel composition

We now propose a composition operation that allows to combine two APAs. We then show how composition and abstraction can collaborate to avoid state-space explosion.

In our theory, the composition operation is parametrized with a set of synchronization actions. This set allows to specify on which actions the two specifications should collaborate and on which actions they can behave individually. The composition of two must transitions is a must transition, but composing a must with a may leads to a may transition.

Definition 11 (Parallel composition of APAs)**.** *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A', L', AP', V', s'_0)$ be APAs and assume $AP \cap AP' = \emptyset$. The parallel composition of N and N' w.r.t. synchronization set $\bar{A} \subseteq A \cap A'$, written as $N \|_{\bar{A}} N'$, is given as $N \|_{\bar{A}} N' = (S \times S', A \cup A', \tilde{L}, AP \cup AP', \tilde{V}, (s_0, s'_0))$ where*

- *$\tilde{L}$ is defined as follows:*
  - *For all $(s, s') \in S \times S'$, $a \in \bar{A}$, if there exists $\varphi \in C(S)$ and $\varphi' \in C(S')$, such that $L(s, a, \varphi) \neq \bot$ and $L'(s', a, \varphi') \neq \bot$, define $\tilde{L}((s, s'), a, \tilde{\varphi}) = L(s, a, \varphi) \sqcap L'(s', a, \varphi')$ with $\tilde{\varphi}$ the new constraint in $C(S \times S')$ such that $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff there exists $\mu \in Sat(\varphi)$ and $\mu' \in Sat(\varphi')$ such that $\tilde{\mu}(u, v) = \mu(u) \cdot \mu'(v)$ for all $u \in S$ and $v \in S'$.*

    If either for all $\varphi \in C(S)$, we have $L(s, a, \varphi) = \bot$, or $\forall \varphi' \in C(S')$, we have $L'(s', a, \varphi') = \bot$ then for all $\tilde{\varphi} \in C(S \times S')$, $\tilde{L}((s, s'), a, \tilde{\varphi}) = \bot$.
  - *For all $(s, s') \in S \times S'$, $a \in A \backslash \bar{A}$, and for all $\varphi \in C(S)$, define $\tilde{L}((s, s'), a, \tilde{\varphi}) = L(s, a, \varphi)$ with $\tilde{\varphi}$ the new constraint in $C(S \times S')$ such that $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff for all $u \in S$ and $v \neq s'$, $\tilde{\mu}(u, v) = 0$ and the distribution $\mu : t \mapsto \tilde{\mu}(t, s')$ is in $Sat(\varphi)$.*
  - *For all $(s, s') \in S \times S'$, $a \in A' \backslash \bar{A}$, and for all $\varphi' \in C(S')$, define $\tilde{L}((s, s'), a, \tilde{\varphi}') = L'(s', a, \varphi')$ with $\tilde{\varphi}'$ the new constraint in $C(S \times S')$ such that $\tilde{\mu'} \in Sat(\tilde{\varphi}')$ iff for all $u \neq s$ and $v \in S'$, $\tilde{\mu'}(u, v) = 0$ and the distribution $\mu' : t' \mapsto \tilde{\mu'}(s, t')$ is in $Sat(\varphi')$.*
- *$\tilde{V}$ is defined as follows: for all $(s, s') \in S \times S'$, $\tilde{V}((s, s')) = \{\tilde{B} = B \cup B' \mid B \in V(s)$ and $B' \in V'(s')\}$.*

Contrary to the conjunction operation, Composition is defined for both dissimilar alphabets and non-deterministic APAs. Since PAs are a restriction of APAs, their compositions is defined in the same way. By inspecting Definition 11, one can see that the composition of two APAs whose constraints are systems of linear inequalities (or polynomial constraints) may lead to an APA whose constraints are polynomial. One can also see that the conjunction of two APAs with polynomial constraints is an APA with polynomial constraints. The class of polynomial constraints APAs is closed under all compositional design operations.

The following theorem characterizes the relation between parallel composition and weak refinement.

Theorem 6. *Given a synchronization set $\bar{A}$, the parallel composition operator $\|_{\bar{A}}$ defined above is a precongruence with respect to weak refinement.*

The fact that abstraction preserves weak refinement (cf. Theorem 4), and that weak refinement is a pre-congruence w.r.t. parallel composition, enables us to apply abstraction in a component-wise manner. That is to say, rather than first generating (the typically large PA) $M\|_{\bar{A}}N$, and then applying abstraction, it allows for first applying abstraction, yielding $\alpha_1(M)$ and $\alpha_2(N)$, respectively, and then constructing $\alpha_1(M)\|_{\bar{A}}\alpha_2(N)$. Possibly a further abstraction of $\alpha_1(M)\|_{\bar{A}}\alpha_2(N)$ can be employed. The next theorem shows that component-wise abstraction is as powerful as applying the combination of the "local" abstractions to the entire model.

Theorem 7. *Let M and N be APA, $\bar{A}$ a synchronization set, and α_1, α_2 be abstraction functions, then:*

$$\alpha_1(M) \parallel_{\bar{A}} \alpha_2(N) = (\alpha_1 \times \alpha_2)(M \parallel_{\bar{A}} N) \qquad \textit{up to isomorphism}$$

The above theorem helps avoiding state-space explosion when combining systems by allowing for abstraction as soon as possible.

6 Completeness and Relation with CMCs

In this section, we propose a class of APAs on which thorough and strong refinements coincide. For doing so, we will compare the expressiveness power of APAs and CMCs, showing that APAs can also act as a specification theory for MCs. We now introduce an important definition that will be used through the rest of the section.

Definition 12. *We say that an APA $N = (S, A, L, AP, V, s_0)$ is in a single valuation normal form iff all its admissible valuations sets are singletons, i.e. for all $s \in S$, we have $|V(s)| = 1$.*

It is worth mentioning that any APA with a single valuation in the initial state can be turned into an APA in single valuation normal form that accepts the same set of implementations (see [123] for such a transformation that preserves determinism).

Some results on CMCs. We recap the definitions of MCs and CMCs. Informally, a MC is a PA with a single probability distribution per state.

Definition 13 (Markov Chain)**.** *$P = \langle Q, q_0, \pi, A, V\rangle$ is a* Markov Chain *if Q is a set of states containing the initial state q_0, A is a set of atomic propositions, $V : Q \to 2^A$ is a state valuation, and $\pi : Q \to (Q \to [0, 1])$ is a probability transition function: $\sum_{q' \in Q} \pi(q)(q') = 1$ for all $q \in Q$.*

We now formally introduce CMC, our abstraction theory for MCs.

Definition 14 (Constraint Markov Chain)**.** *A* Constraint Markov Chain *is a tuple $C = \langle Q, q_0, \psi, AP, V\rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\psi : Q \to (Dist(Q) \to \{0, 1\})$ is a constraint function, AP is a set of atomic propositions and $V : Q \to 2^{2^{AP}}$ is a state labeling function.*

For each state $q \in Q$, the constraint function ψ is such that, for all distribution π on Q, $\psi(q)(\pi) = 1$ iff the distribution π is allowed in state q.

We say that a CMC C is deterministic iff for all states $q, q', q'' \in Q$, if there exists $\pi' \in Dist(Q)$ such that $(\psi(q)(\pi') \wedge (\pi'(q') \neq 0))$ and $\pi'' \in Dist(Q)$ such that $(\psi(q)(\pi'') \wedge (\pi''(q'') \neq 0))$, then we have that $V(q') \cap V(q'') = \emptyset$. Single valuation normal form of CMCs is defined similarly as for APAs. The satisfaction relation between MCs and CMCs as well as the notions of weak and strong refinements are also defined similarly as for APAs. We will use the following result.

Theorem 8 ([63])**.** *For deterministic CMCs in single valuation normal form, strong refinement coincides with thorough and weak refinement.*

On the relation between CMCs and APAs. We now show that APAs can act as a specification theory for MCs. For doing so, we propose a satisfaction relation between MCs and APAs. Our definition is in two steps. First we show how to use PAs as a specification theory for MCs. Then, we use the existing relation between PAs and APAs to conclude.

Definition 15. *Let $P = (S, A, L, AP, V, s_0)$ be a PA with $A \cap AP = \emptyset$. Let $M = \langle Q, q_0, \pi, A_M, V_M \rangle$ be a bipartite Markov chain such that (1) $Q = Q_N \cup Q_D$, with $Q_N \cap Q_D = \emptyset$, for all $q, q' \in Q_N, \pi(q, q') = 0$ and for all $q, q' \in Q_D$, $\pi(q)(q') = 0$, (2) $q_0 \in Q_D$, and (3) $A_M = A \cup AP$. Let $\mathcal{R} \subseteq Q_D \times S$. $\mathcal{R}$ is a satisfaction relation iff whenever $q \mathcal{R} s$, we have*

1. *$V_M(q) = V(s)$.*
2. *For all action $a \in A$ and distribution μ over S such that $L(s, a, \mu) = \top$, there exists $q' \in Q_N$ such that $V(q') = V(s) \cup \{a\}$, $\pi(q)(q') > 0$, and $\pi(q') \Subset_{\mathcal{R}} \mu$.*
3. *For all state $q' \in Q_N$ such that $\pi(q, q') > 0$, there exists an action $a \in A$ and a distribution μ over S such that $V(q') = V(s) \cup \{a\}$, $L(s, a, \mu) = \top$, and $\pi(q') \Subset_{\mathcal{R}} \mu$.*

We say that M satisfies P iff there exists a satisfaction relation $\mathcal{R}$ such that $q_0 \mathcal{R} s_0$.

The satisfaction relation between MCs and APAs follows directly. We say that a MC M satisfies an APA N, which we write $M \models_{MC} N$, iff there exists a PA P such that M satisfies P and P satisfies N.

Expressivity Completeness. In the previous section, we have proposed a satisfaction relation for MCs with respect to APAs. We now propose the following theorem that relates the expressive power of CMCs and APAs.

Theorem 9. *Let $N = (S, A, L, AP, V, s_0)$ be a deterministic APA in single valuation normal form and such that $AP \cap A = \emptyset$. There exists a deterministic CMC $\widehat{N}$ in single valuation normal form such that for all MC M, $M \models_{MC} N \iff M \models \widehat{N}$.*

We have just shown that for all APA N, there exists a CMC $\widehat{N}$ such that $[\![N]\!]_{MC} = [\![\widehat{N}]\!]$. The reverse of the theorem also holds up to a syntactical transformation that preserves sets of implementations (see [123] for details). This result together with Theorem 8 leads to the following important result.

Theorem 10. *For deterministic APAs with single valuations in the initial state, strong refinement coincides with thorough and weak refinement.*

7 Conclusion

This paper presents a novel abstraction for PAs and proposes the first specification theory for them. In addition, the paper also studies the relation between abstraction and compositional design in combating the state-space explosion problem.

There are various directions for future research. The first of them being to implement and evaluate our results. This would require to design efficient algorithms for the compositional design operators. Also, it would be of interest to embed our abstraction procedure in a CEGAR model checking algorithm. Another interesting direction would be to design an algorithm to decide thorough refinement and characterize the complexity of this operation. Finally, one should also consider a continuous-timed extension of our model inspired by [49].

Paper E – New Results on Abstract Probabilistic Automata

Benoit Delahaye
INRIA/IRISA, Rennes

Joost-Pieter Katoen
RWTH Aachen University

Kim G. Larsen
Aalborg University

Axel Legay
INRIA/IRISA, Rennes

Mikkel L. Pedersen
Aalborg University

Falak Sher
RWTH Aachen University

Andrzej Wasowski
IT University, Copenhagen

1 Abstract

Probabilistic Automata (PAs) are a recognized framework for modeling and analysis of nondeterministic systems with stochastic behavior. Recently, we proposed Abstract Probabilistic Automata (APAs)—an abstraction framework for PAs. In this paper, we discuss APAs over dissimilar alphabets, a determinisation operator, conjunction of non-deterministic APAs, and an APA-embedding of Interface Automata. We conclude introducing a tool for automatic manipulation of APAs.

2 Introduction

Probabilistic Automata (PAs), proposed by Segala [52], are a mathematical framework for rigorous specification and analysis of non-deterministic probabilistic systems, or more precisely systems that combine concurrent behaviour with discrete probabilistic choice. PAs are akin to Markov decision processes (MDPs). A detailed comparison with models such as MDPs, as well as generative and reactive probabilistic transition systems

is given in [116]. PAs are recognized as an adequate formalism for various applications including randomized distributed algorithms and fault tolerant systems [117, 118, 119, 120, 122].

Recently [64], we have proposed Abstract Probabilistic Automata (APAs), that is a compact abstraction formalism for sets of PAs. The model is a marriage between our new abstract model for Markov chains [63] and modal automata, an abstraction for non-deterministic systems promoted by [42] and [41]. In an APA, non-deterministic behaviors are typed with may and must modalities. The must modalities identify those behaviors that must be present in any implementation, while the may modalities refer to those behaviors that are allowed to be omitted in an implementation. In APAs, probability distributions that govern the successor states are replaced by set of distributions, each of them representing a possible implementation of the abstraction.

One of the major contributions of [64] was to develop the first specification theory for PAs. This includes a satisfaction relation (to decide whether a PA is an implementation of an APA), a consistency check (to decide whether the specification admits an implementation), a refinement (to compare specifications in terms of inclusion of sets of implementations), logical composition (to compute the intersection of sets of implementations), and structural composition (to combine specifications). Our framework also supports incremental design [91]. In addition, we have proposed an abstraction mechanism that allows to simplify the design in an aggressive manner.

While the theory is already quite complete, some fundamental aspects have to be improved in order to make it attractive from a design point of view. First, our theory assumes that non-stochastic behaviors of the components are defined over the same alphabets. In various contexts this assumption is unrealistic. Indeed, one should be able to combine the existing design with new components whose ports and variables are not yet specified [36].

Second, the conjunction operation has only been defined for those systems whose non-stochastic behaviors are described in a deterministic manner. Again, from the practical point of view, one should be capable of handling non-determinism inherent to transition systems and concurrency.

Third, the existing composition operator for APAs assumes closed system composition, inhibiting reasoning about open systems. Support for open systems, enables incremental modeling and allows reasoning not only about stochastic components, but also about the requirements for their usage (environment).

The aim of this paper is to propose solutions to the above mentioned problems. Our contributions are described below.

- We extend the theory of APAs to support specifications over dissimilar alphabets. The principle is similar to what has been proposed for modal automata in [42]. Unfortunately due to interweaving of probabilistic and non-deterministic choices, proofs of correctness of [42] could not be reused.

- We show that the definition of conjunction proposed in [64] is too strong for non-deterministic APAs. We propose a more general construction that corresponds to

the greatest lower bound with respect to a new refinement relation, more precise than refinements introduced before [64].

This result is of additional theoretical interest. In [44] we have shown that such greatest lower bound generally does not exist for modal automata. Nevertheless it was possible to introduce it for APAs, which contain modal automata. The new construction works for modal automata encoded as APAs, because it potentially produces an APA which is not an encoding of any modal automaton.

- We present a determinization algorithm that given an APA whose non-stochastic behaviors are non-deterministic, computes its deterministic abstraction. We also show that there are APAs for which there exists no deterministic APAs accepting the same set of models (so determinization must be lossy). This lossiness of the abstraction further motivates the need for the weaker conjunction operator mentioned above.

- We propose a translation of APAs to Abstract Probabilistic Interfaces (API) that is a stochastic extension of the classical game-based interface automata proposed by de Alfaro et al. APIs are similar to the stochastic I/O automata of Lynch except that they encompass a game-based semantics that allows for an optimistic composition. Given two APIs, one can compute the environment in where they can work together in a proper manner.

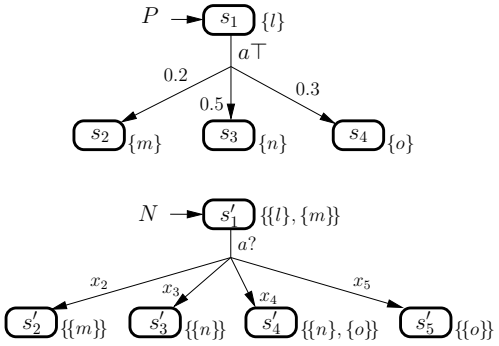
- We introduce the APAC tool, in which the APA theory has been implemented. APAC relies on the SMT solver Z3 [112] for checking relations between the probability distributions of the components. To the best of our knowledge, this is the first implementation of a theory that proposes both logical and structural compositions for Probabilistic Automata.

3 Background

We now briefly introduce the specification theory of *Abstract Probabilistic Automata* as presented in [64]. We begin with the notion of a probabilistic automaton [52]. Let $\mathrm{Dist}(S)$ denote a set of all discrete probability distributions over a finite set S, and let $\mathbb{B}_2 = \{\top, \bot\}$. Then:

Definition 1. *A* probabilistic automaton *(PA) is a tuple* (S, A, L, AP, V, s_0)*, where* S *is a finite set of states with the initial state* $s_0 \in S$*,* A *is a finite set of actions,* L*:* $S \times A \times \mathrm{Dist}(S) \to \mathbb{B}_2$ *is a (two-valued transition) function,* AP *is a finite set of atomic propositions and* V*:* $S \to 2^{AP}$ *is a state-labeling function.*

For a state s, an action a and a probability distribution μ, the value of $L(s, a, \mu)$ symbolizes the presence ($\top$) or absence ($\bot$) of a transition from s under action a to a distribution μ specifying possible successor states. In practice L may be a partial



$\varphi_x \equiv (x_2{+}x_3 \geq 0.7) \wedge (x_4{+}x_5 \geq 0.2) \wedge (x_2{+}x_3{+}x_4{+}x_5 = 1)$

Figure 1: Examples of a PA (top) and of an APA (bottom)

function—if a value of L is unspecified for a given combination of arguments, then it behaves as if it was specified to be $\bot$.

Example 1. *The top of Figure 1 shows a PA P over the singleton set of actions $A = \{a\}$ and atomic propositions $AP = \{l, m, n, o\}$. In the figure, $\top$-transitions are drawn explicitly, and $\bot$-transitions are elided. For example, in P there is one a-transition from s_1 to the distribution* $[0, 0.2, 0.5, 0.3]$.

An abstract probabilistic automaton relaxes the above definition to allow describing multiple probabilistic automata (including their probability distributions) within a single abstraction. Let $C(S)$ denote a set of all constraints over discrete probability distributions over a finite set S; so that each element $\varphi \in C(S)$ describes a set of distributions: $\mathit{Sat}(\varphi) \subseteq \mathrm{Dist}(S)$. In this paper we do not fix the language of constraints used to generate $C(S)$. Instead, we just require that $C(S)$ is closed under usual Boolean connectives, that it includes equalities over summations and multiplications of probability values, and that it allows for existential quantification of variables. Also let $\mathbb{B}_3 = \{\top, ?, \bot\}$. Then:

Definition 2. *An* Abstract Probabilistic Automaton *(APA) is a tuple (S, A, L, AP, V, s_0), where S is a finite set of states, $s_0 \in S$, A is a finite set of actions, and AP is a finite set of atomic propositions. $L : S \times A \times C(S) \to \mathbb{B}_3$ is a* three-*valued distribution-constraint function, and $V : S \to 2^{2^{AP}}$ maps each state in S to a set of admissible labelings.*

APAs play the role of specifications in our framework. An APA transition abstracts transitions of a certain unknown PA, called its implementation. Given a state s, an action a, and a constraint φ, the value of $L(s, a, \varphi)$ gives the modality of the transition. More precisely the value $\top$ means that transitions under a must exist in the PA to every distribution in $\mathit{Sat}(\varphi)$; ? means that these transitions are permitted to exist; $\bot$

means that such transitions must not exist. Again L may be partial. In practice, as will be seen in later definitions, a lack of value for given argument is equivalent to the $\bot$ value, so we will sometimes avoid defining $\bot$-value rules in constructions to avoid clutter, and occasionally will say that something applies if L takes the value of $\bot$, meaning that it is either taking this value or it is undefined. The function V labels each state with a subset of the powerset of AP, which models a disjunctive choice of possible combinations of atomic propositions.

We occasionally write $\mathsf{Must}(s)$ for the set of all actions a such that there exists φ, so that $L(s, a, \varphi) = \top$, and write $\mathsf{May}(s)$ for the set of all actions b such that there exists ψ, so that $L(s, b, \psi) = ?$.

Example 2. *An example of an APA N, with the same signature as the PA P, is shown in the bottom of Figure 1. Again we follow a graphical convention of eliding $\bot$-transitions. Must ($\top$) and may ($?$) transitions are shown explicitly with modalities appended to the action label. Here, in the APA N, there is one allowed a-transition from N to a constraint φ_x (specified under the automaton).*

A PA is essentially an APA in which every transition $L(s, a, \mu) = m$ is represented by the same modality transition $L(s, a, \varphi) = m$ with $Sat(\varphi) = \{\mu\}$, and each state-label consists of a single set of propositions.

As already mentioned, we relate APA specifications to PAs implementing them, by extending the definitions of satisfaction introduced in [46]. We begin by relating distributions between sets of states [64]:

Definition 3. *Let S and S' be non-empty sets, and μ, μ' be distributions; $\mu \in \mathrm{Dist}(S)$ and $\mu' \in \mathrm{Dist}(S')$. We say that μ is* simulated *by μ' with respect to a relation $R \subseteq S \times S'$ and a* correspondance function *$\delta : S \to (S' \to [0, 1])$ iff*

1. *For all $s \in S$, $\delta(s)$ is a distribution on S' if $\mu(s) > 0$*
2. *For all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$,*
3. *Whenever $\delta(s)(s') > 0$ then $(s, s') \in R$.*

We write $\mu \Subset_R^\delta \mu'$ meaning that μ is simulated by μ' with respect to R and δ, and we write $\mu \Subset_R \mu'$ iff there exists a function δ such that $\mu \Subset_R^\delta \mu'$.

Now, the following definition, originating in [64], formally establishes the roles of PAs and APAs as implementations and specifications respectively. For a PA P satisfying an APA N we require that any must-transition of N is matched by a must-transition of P agreeing with the distributions specified by the constraint, and any must-transition of P is matched by a may- or must-transition in N.

Definition 4. *Let $P = (S, A, L, AP, V, s_0)$ be a PA and $N = (S', A, L', AP, V', s'_0)$ be an APA. A binary relation $R \subseteq S \times S'$ is a* satisfaction relation *iff, for any $(s, s') \in R$, the following conditions hold:*

1. *Whenever $L'(s', a, \varphi') = \top$ for some $a \in A$, $\varphi' \in C(S')$ then also $L(s, a, \mu) = \top$ for some distribution μ such that $\mu \Subset_R \mu'$ and $\mu' \in Sat(\varphi')$.*
2. *Whenever $L(s, a, \mu) = \top$ for some $a \in A$, $\mu \in \mathrm{Dist}(S)$ then $L(s', a, \varphi')$ is defined with $L'(s', a, \varphi') \neq \bot$ for some $\varphi' \in C(S')$ and $\mu' \in Sat(\varphi')$ such that $\mu \Subset_R \mu'$.*
3. *$V(s) \in V'(s')$.*

We say that P satisfies N, denoted $P \models N$, iff there exists a satisfaction relation relating s_0 and s'_0. If $P \models N$, then P is called an implementation of (specification) N.

Example 3. *The relation $\mathcal{R} = \{(s_1 s'_1), (s_2 s'_2), (s_3 s'_3), (s_4 s'_4), (s_4 s'_5)\}$ is a satisfaction relation between P and N of Fig. 1. It is easy to see that all pairs in $R \setminus \{(s_1, s'_1)\}$ fulfill the definition, as they have no outgoing transitions and the labelings of states in P respect the labeling constraints of N.*

So consider (s_1, s'_1). Condition 2 is satisfied vacuously. Take $\mu' = [0, 0.2, 0.5, 0.15, 0.15] \in Sat(\varphi_x)$. Let $\mu = [0, 0.2, 0.5, 0.3]$ be the distribution of the only a-transition of P. We are showing that condition 1 above is satisfied, i.e. that $\mu \Subset_R \mu'$. This is witnessed by the following correspondance function: $\delta(s_2, s'_2) = \delta(s_3, s'_3) = 1$, $\delta(s_4, s'_4) = \delta(s_5, s'_5) = 0.5$, and $\delta(s_i, s'_j) = 0$ for all remaining pairs of states. □

We denote the set of all implementations of N by $[\![N]\!] = \{P \mid P \models N\}$. An APA N is said to be *consistent* iff $[\![N]\!] \neq \emptyset$. A state s of an APA is called consistent if and only if $V(s) \neq \emptyset$ and $(L(s, a, \varphi) = \top \implies Sat(\varphi) \neq \emptyset)$. If all states of N are consistent then N is consistent, but not necessarily the other way around.

In [64], a pruning operator β is defined that filters out distributions leading to inconsistent states, making these states unreachable. After a single application of β to an APA N, it holds that $[\![N]\!] = [\![\beta(N)]\!]$. Pruning itself may introduce inconsistent states, so we apply β until a fixpoint is reached, which is guaranteed to happen after a finite number of steps.

We say that an APA N *thoroughly refines* another APA M iff $[\![N]\!] \subseteq [\![M]\!]$. Such notion of refinement, although theoretically satisfying, is not easy to establish algorithmically. For this reason [64] introduces a more syntactic refinement, called a weak refinement:

Definition 5. *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP, V', s'_0)$ be APAs. A binary relation $R \subseteq S \times S'$ is a* weak refinement *relation iff, for all $(s, s') \in R$, the following conditions hold:*

1. *Whenever* $L'(s', a, \varphi') = \top$ *for some action* $a \in A$ *and distribution constraint* $\varphi' \in C(S')$ *then* $L(s, a, \varphi) = \top$ *for some distribution constraint* $\varphi \in C(S)$ *such that* $\forall \mu \in Sat(\varphi).\, \exists \mu' \in Sat(\varphi').\, \mu \Subset_R \mu'$
2. *Whenever* $L(s, a, \varphi) \neq \bot$ *for some* $a \in A$ *and* $\varphi \in C(S)$ *then* $L'(s', a, \varphi') \neq \bot$ *for some constraint* $\varphi' \in C(S')$ *such that* $\forall \mu \in Sat(\varphi).\, \exists \mu' \in Sat(\varphi').\, \mu \Subset_R \mu'$
3. $V(s) \subseteq V'(s')$.

We say that N *weakly refines* N'*, denoted* $N \preceq N'$*, iff there exists a weak refinement relation relating* s_0 *and* s'_0.

The correspondence function δ is not fixed in advance, and can be chosen for each μ and μ' separately so that $\mu \Subset^{\delta}_{\mathcal{R}} \mu'$. The weak refinement is sound with respect to the thorough refinement: if $N \preceq N'$ then $[\![N]\!] \subseteq [\![N']\!]$ [64]. It is known that the two refinements coincide for deterministic APAs if the initial state admits exactly one labeling: $(|V(s_0) \models 1)$ [64].

Definition 6. *An APA* $N = (S, A, L, AP, V, s_0)$ *is deterministic if it satisfies the following two conditions:*

[action-determinism] *An action determines the successor:* $\forall s \in S. \forall a \in A. |\{\varphi \in C(S) \mid L(s, a, \varphi) \neq \bot\}| \leq 1$.

[labeling-determinism] *Labels discern possible successor states:* $\forall s \in S. \forall a \in A. \forall \varphi \in C(S)$ *if* $L(s, a, \varphi) \neq \bot$ *then:*

$$\forall \mu', \mu'' \in Sat(\varphi), s', s'' \in S.$$
$$\left(\mu'(s') > 0 \wedge \mu''(s'') > 0 \;\Rightarrow V(s') \cap V(s'') = \emptyset\right).$$

Example 4. *The APA* N *in Fig. 1 is action-deterministic, but not labeling-deterministic. The distributions* $\mu' = [0, 0.4, 0.4, 0.1, 0.1]$*,* $\mu'' = [0, 0.5, 0.2, 0.3, 0)$ *are both in* $Sat(\varphi_x)$ *and give positive probability to* s'_3 *and* s'_4*, respectively, while their labeling constraints intersect on* $\{\{n\}\}$.

To conclude this section, we present the definition of parallel composition, which is known to be a precongruence with respect to weak refinement [64].

Definition 7. *Let* $N = (S, A, L, AP, V, s_0)$ *and* $N' = (S', A', L', AP', V', s'_0)$ *be APAs with* $AP \cap AP' = \emptyset$*. The parallel composition of* $N \|_{\bar{A}} N'$ *with respect to a synchronization set* $\bar{A} \subseteq A \cap A'$ *is given as* $N \|_{\bar{A}} N' = (S \times S', A \cup A', \tilde{L}, AP \cup AP', \tilde{V}, (s_0, s'_0))$ *and*

1. *For each* $a \in \bar{A}$

$$\frac{\exists \varphi.\, L(s, a, \varphi) \neq \bot \qquad \forall \varphi'.\, L'(s', a, \varphi') \neq \bot}{\tilde{L}((s, s'), a, \tilde{\varphi}) = L(s, a, \varphi) \sqcap L'(s', a, \varphi')} \tag{1}$$

$$\frac{\forall \varphi.\, L(s, a, \varphi) = \bot \vee \forall \varphi'.\, L(s', a, \varphi') = \bot}{\forall \tilde{\varphi}'. \tilde{L}((s, s'), a, \tilde{\varphi}') = \bot} \tag{2}$$

where $\tilde{\varphi} \in C(S \times S')$ *is so that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff there exists* $\mu \in Sat(\varphi)$ *and* $\mu' \in Sat(\varphi')$ *such that* $\tilde{\mu}(u,v) = \mu(u) \cdot \mu'(v)$ *for all* $u \in S$ *and* $v \in S'$.

2. *For each* $a \in A \setminus A'$*:*

$$\frac{Sat(\tilde{\varphi}) = \{\tilde{\mu} \mid \tilde{\mu}(\cdot, s') \in Sat(\varphi), \tilde{\mu}(u,v) = 0 \text{ for } v \neq s'\}}{\tilde{L}((s,s'),a,\tilde{\varphi}) = L(s,a,\varphi)}$$

3. *And symmetrically for each* $a \in A' \setminus A$*:*

$$\frac{Sat(\tilde{\varphi}') = \{\tilde{\mu}' \mid \tilde{\mu'}(s, \cdot) \in Sat(\varphi'), \tilde{\mu'}(u,v) = 0 \text{ for } u \neq s\}}{\tilde{L}((s,s'),a,\tilde{\varphi}') = L'(s',a,\varphi')}$$

4. $\tilde{V}((s,s')) = \{B \cup B' \mid B \in V(s) \text{ and } B' \in V'(s')\}$.

4 Extensions of Alphabets

So far, the specification theory of APAs has required that all specifications share same alphabets of actions and labels. We are now going to lift this restriction, by introducing the alphabet extension mechanism. Just like for modal transition systems [35], for which there exist two ways of extending signatures [36], for APAs it is also necessary to choose the modality of transitions for new actions introduced, depending on the operation being applied to the result.

The weak extension is used when conjoining specifications with different signatures. This extension adds may loop transitions for all new actions and extends the sets of atomic propositions in a classical way:

Definition 8. *Let* $N = (S, A, L, AP, V, s_0)$ *be an APA, and let* A' *and* AP' *be sets of actions and atomic propositions such that* $A \subseteq A'$ *and* $AP \subseteq AP'$*. Let the weak extension of* N *to* (A', AP') *be the APA* $N \Uparrow (A', AP') = (S, A', L', AP', V', s_0)$ *such that for all states* $s \in S$*:*

- $L'(s,a,\varphi) = L(s,a,\varphi)$ *if* $a \in A$,
- $L'(s,a,\varphi) = ?$ *if* $a \in A' \setminus A$ *and* φ *only admits a single point distribution* μ *such that* $\mu(s) = 1$.
- $V'(s) = \{B \subseteq AP' \mid B \cap AP \in V(s)\}$.

A different extension, the strong one, is used in parallel composition. This extension adds must self-loops for all new actions and extends the sets of atomic propositions in a classical way. See [123] for a formal definition.

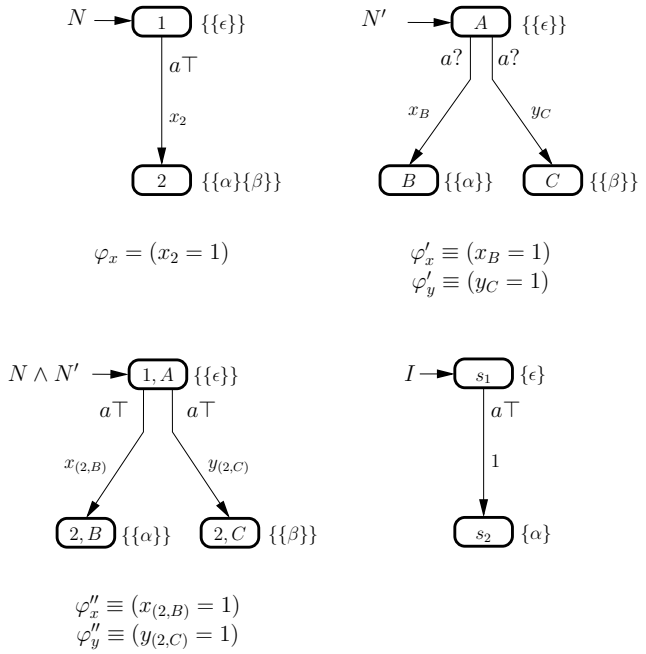


Figure 2: Illustration that conjunction using Definition 9 is not a greatest lower bound.

These different notions of extension give rise to different notions of satisfaction and refinement between structures with dissimilar sets of actions. Satisfaction (or refinement) between structures with different sets of actions is defined as the satisfaction (respectively refinement) between the structures after extension to a union of signatures.

5 Conjunction

5.1 Incompleteness of Conjunction

A conjunction operator combines two specifications into a single one, ideally describing the intersection of their implementation sets (so $[\![N \wedge M]\!] = [\![N]\!] \cap [\![M]\!]$). In [64], conjunction was only defined for action-deterministic APAs with *identical alphabets*. In this paper, we first show that construction is incorrect for non-deterministic APAs. Then we generalize it to the non-deterministic case *with dissimilar alphabets*. Let's recall the definition given in [64]:

Definition 9. *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP, V', s'_0)$ be action-deterministic APAs. Their conjunction is the APA $N \wedge N' = (S \times S', A, \tilde{L}, AP, \tilde{V}, (s_0, s'_0))$ where $\tilde{V}((s, s')) = V(s) \cap V'(s')$ and $\tilde{L}$ is defined as follows. Given an action $a \in A$ and a state $(s, s') \in S \times S'$:*

$$\frac{\exists\varphi.\, L(s,a,\varphi) = \top \qquad \forall\varphi'.\, L'(s',a,\varphi') = \bot}{\tilde{L}((s,s'),a,\mathit{false}) = \top} \tag{3}$$

$$\frac{\forall\varphi.\, L(s,a,\varphi) = \bot \qquad \exists\varphi'.\, L'(s',a,\varphi') = \top}{\tilde{L}((s,s'),a,\mathit{false}) = \top} \tag{4}$$

$$\frac{\forall\varphi.\, L(s,a,\varphi) \neq \top \qquad \forall\varphi'.\, L'(s',a,\varphi') = \bot}{\tilde{L}((s,s'),a,\_) = \bot} \tag{5}$$

$$\frac{\forall\varphi.\, L(s,a,\varphi) = \bot \qquad \forall\varphi'.\, L'(s',a,\varphi') \neq \top}{\tilde{L}((s,s'),a,\_) = \bot} \tag{6}$$

$$\frac{L(s,a,\varphi) \neq \bot \qquad L'(s',a,\varphi') \neq \bot}{\tilde{L}((s,s'),a,\tilde{\varphi}) = L(s,a,\varphi) \sqcup L'(s',a,\varphi')} \tag{7}$$

where $\tilde{\varphi} \in C(S \times S')$ *such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff both distribution* $\mu : t \to \sum_{t' \in S'} \tilde{\mu}((t,t'))$ *is in* $Sat(\varphi)$ *and distribution* $\mu' : t' \to \sum_{t \in S} \tilde{\mu}((t,t'))$ *is in* $Sat(\varphi')$.

In [64] it is shown that this construction captures the greatest lower-bound with respect to weak refinement, i.e. For N, N', and N'' action-deterministic consistent APAs over the same action alphabet we have that:

- $\beta^*(N \wedge N') \preceq N$ and $\beta^*(N \wedge N') \preceq N'$
- If $N'' \preceq N$ and $N'' \preceq N'$ then $N'' \preceq \beta^*(N \wedge N')$.

At the same time this construction is inadequate for non-deterministic APAs. Combining one must-transition with several may-transitions using the same action is problematic. We show that the conjunction of non-deterministic APAs, using the definition above, is not a lower bound with respect to neither thorough refinement nor weak refinement.

Lemma 1. *The construction of Def. 9 is strictly stronger than the greatest lower bound of the thorough refinement and of the weak refinement for non-deterministic APAs, in the following sense:*

1. *There exists a PA* I*, and APAs* N *and* N' *such that* $I \models N$ *and* $I \models N'$ *but not* $I \models N \wedge N'$.
2. *There exists an APA* M*, such that* $M \preceq N$ *and* $M \preceq N'$ *but not* $M \preceq N \wedge N'$.

Proof. Figure 2 presents two APAs $N = (\{1,2\}, \{a\}, L, \{\epsilon, \alpha, \beta\}, V, 1)$ and $N' = (\{A, B, C\}, \{a\}, L', \{\epsilon, \alpha, \beta\}, V', A)$ together with their conjunction $N \wedge N'$, constructed according to Definition 9. The rightmost part of the figure shows a PA $I = (\{s_1, s_2\}, \{a\}, L_I, \{\epsilon, \alpha, \beta\}, V_I, s_1)$, which shows that the conjunction is too strong with respect to the thorough refinement. It holds that $I \models N$ and $I \models N'$, but $I \not\models (N \wedge N')$. The conjunction of N and N' has two must transitions, and the one leading to $(2, C)$ is not fulfilled by I.

For the second part of the theorem it is sufficient to interpret I as the APA M and the argument follows. □

For dissimilar alphabets, conjunction can be treated separately from alphabet extension. One first computes the (weak) alphabet extensions for both APAs, and then compute conjunction using the above definition 9. Formally:

Definition 10. *Let $N_1 = (S_1, A_1, L_1, AP_1, V, s_1)$, $N_2 = (S_2, A_2, L_2, AP_2, V_2, s_2)$ be action-deterministic APAs. Their conjunction is the APA $N_1 \wedge N_2 = [N_1 \Uparrow \alpha] \wedge [N_2 \Uparrow \alpha]$, with $\alpha = (A_1 \cup A_2, AP \cup AP')$ and $\wedge$ defined as above.*

Considering the above definition for APAs with dissimilar action sets, the following theorem trivially holds.

Theorem 2. *Let N_1, N_2, and N_3 be action-deterministic consistent APAs over action alphabets A_1, A_2, A_3 and atomic proposition sets AP_1, AP_2 and AP_3 respectively. Let $\alpha_{ij} = (A_i \cup A_j, AP_i \cup AP_j)$, and $\alpha_{123} = (\bigcup_{i=1}^{3} A_i, \bigcup_{i=1}^{3} AP_i)$. Then:*

1. $\beta^*(N_1 \Uparrow \alpha_{12} \wedge N_2 \Uparrow \alpha_{12}) \preceq N_1 \Uparrow \alpha_{12}$
2. *If $N_3 \Uparrow \alpha_{123} \preceq N_1 \Uparrow \alpha_{123}$ and $N_3 \Uparrow \alpha_{123} \preceq N_2 \Uparrow \alpha_{123}$ then $N_3 \Uparrow \alpha_{123} \preceq \beta^*(N_1 \Uparrow \alpha_{12} \wedge N_2 \Uparrow \alpha_{12}) \Uparrow \alpha_{123}$*

5.2 Weak weak Refinement

The weak refinement (Def. 5), along with the so called *strong refinement* [64], had been introduced for Constraint Markov Chains in [63], as syntax directed sound characterizations of thorough refinement. They were then generalized to APAs in [64] in a "natural" way.

As we see from Lemma 1 the conjunction construction of [64] is too strong with respect to the weak refinement for non-deterministic systems. In order to address this problem, one can (potentially) either weaken the construction or strenghten the refinement. There are issues with any of the solutions. First, strengthening the refinement makes it even more strong with respect to thorough refinement (so it becomes less precise which is undesirable), and moreover the known strong refinement [64] still violates Lemma 1 (i.e. it is too coarse).

Second, the natural weakening of the construction gives a resulting conjunction APA that is too weak with respect to the weak refinement.

Instead of fine tuning the construction, which could become very complicated, we decided to explore another possibility: namely propose a weaker and more precise refinement, the *weak weak refinement*, which is designed with APAs (and not CMCs) in mind. The weak weak refinement approximates thorough refinement even better than weak refinement, and it has a naturally characterized greatest lower bound.

In the weak refinement, cf. Def. 5, the correspondence function is established for two constraints: for each solution of one, there must be a correspondance to some solution of the other constraints. Weak weak refinement weakens this condition by allowing to choose, for each solution of the first constraint, both a different correspondence function *and* a different constraint (transition) to which it will be linked:

Definition 11. *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A', L', AP', V', s'_0)$ be APAs with $AP = AP'$ and $A = A'$. A relation $R \subseteq S \times S'$ is a* weak weak refinement relation*, iff for all $(s, s') \in R$, the following conditions hold:*

1. $\forall a \in A'. \forall \varphi' \in C(S'). L'(s', a, \varphi') = \top \implies \exists \varphi \in C(S). L(s, a, \varphi) = \top$ *and* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *such that* $\mu \Subset_R \mu'$,
2. $\forall a \in A. \forall \varphi \in C(S). L(s, a, \varphi) \neq \bot \implies \forall \mu \in Sat(\varphi). \exists \varphi' \in C(S'). L'(s', a, \varphi') \neq \bot$ *and* $\exists \mu' \in Sat(\varphi')$ *such that* $\mu \Subset_R \mu'$, *and*
3. $V(s) \subseteq V'(s')$.

We say that N_1 weakly weakly refines N_2, denoted $N_1 \preceq_W N_2$, iff there exists a weak weak refinement relation relating s_0 and s'_0.

It follows directly that weak weak refinement is weaker than weak refinement and thus strong refinement. For action-deterministic APAs, weak weak refinement is equivalent to weak refinement.

5.3 Conjunction of Non-deterministic APAs

We thus propose the following definition for conjunction of possibly non-deterministic APAs.

Definition 12. *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP, V', s'_0)$ be APAs sharing action and proposition sets. Their conjunction $N \owedge N'$ is the APA $(S \times S', A \cup A', \tilde{L}, AP \cup AP', \tilde{V}, (s_0, s'_0))$ where $\tilde{V}((s, s')) = V(s) \cap V'(s')$ and*

$$\frac{a \in (\mathsf{Must}(s') \setminus \mathsf{May}(s)) \cup (\mathsf{Must}(s) \setminus \mathsf{May}(s'))}{\tilde{L}((s, s'), a, \mathit{false}) = \top}, \tag{8}$$

$$\frac{a \in (\mathsf{May}(s) \setminus \mathsf{May}(s')) \cup (\mathsf{May}(s') \setminus \mathsf{May}(s))}{\tilde{L}((s, s'), a, \tilde{\varphi}) = \bot}, \tag{9}$$

$$\frac{a \in \mathsf{May}(s) \cap \mathsf{May}(s') \qquad L(s, a, \varphi) \neq \bot \qquad L'(s', a, \varphi') \neq \bot}{\tilde{L}((s, s'), a, \tilde{\varphi}) = ?}, \tag{10}$$

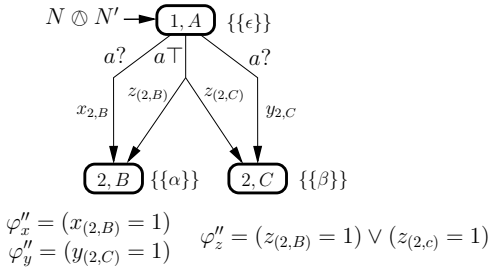


Figure 3: APA $N \oslash N'$ obtained using Definition 12

where $\tilde{\varphi} \in C(S \times S')$ *such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff both distribution* $\mu : t \to \sum_{t' \in S'} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi)$ *and distribution* $\mu' : t' \to \sum_{t \in S} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi')$.

$$\frac{a \in \mathsf{Must}(s) \qquad L(s, a, \varphi) = \top}{\tilde{L}((s, s'), a, \tilde{\varphi}^{\top}) = \top}, \tag{11}$$

where $\tilde{\varphi}^{\top} \in C(S \times S')$ *such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff both the distribution* $\mu : t \to \sum_{t' \in S'} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi)$, *and there exists* $\varphi' \in C(S')$ *with* $L'(s', a, \varphi') \neq \bot$ *and the distribution* $\mu' : t' \to \sum_{t \in S} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi')$.

$$\frac{a \in \mathsf{Must}(s') \qquad L'(s', a', \varphi') = \top}{\tilde{L}((s, s'), a, \tilde{\varphi'}^{\top}) = \top}, \tag{12}$$

where $\tilde{\varphi}^{\top} \in C(S \times S')$ *is such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff both there exists* $\varphi \in C(S)$ *such that* $L(s, a, \varphi) \neq \bot$ *and the distribution* $\mu : t \to \sum_{t' \in S'} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi)$, *and the distribution* $\mu' : t' \to \sum_{t \in S} \tilde{\mu}((t, t'))$ *is in* $Sat(\varphi')$.

The apparent complexity of the new definition concurs our experience from specification theories for discrete systems. For example, in [82] the notion of conjunction is presented for nondeterministic Modal Automata, resulting in a similar sophistication for resolving nondeterminism (modal automata do not contain the probabilistic part).

Example 5. *Following the example of Fig. 2, we build the conjunction of APAs* N *and* N' *using Definition 12. The APA* $N \oslash N'$ *is given in Figure 3. Clearly, the PA* I *given in Figure 2 is an implementation of* $N \oslash N'$.

We now give the main result of the section: as expected, the conjunction operator, given in Definition 12 matches the greatest lower bound of the weak weak refinement.

Theorem 3. *Let N_1, N_2, and N_3 be consistent APAs sharing action and atomic proposition sets. It holds that*

- $\beta^*(N_1 \oslash N_2) \preceq_W N_1$ *and* $\beta^*(N_1 \oslash N_2) \preceq_W N_2$.
- *If* $N_3 \preceq_W N_1$ *and* $N_3 \preceq_W N_2$ *then* $N_3 \preceq_W \beta^*(N_1 \oslash N_2)$.

As expected, the new conjunction is weaker than the old one, thus it gives a more precise result:

Theorem 4. *Let N_1 and N_2 be APAs. It holds that $N_1 \wedge N_2 \preceq N_1 \oslash N_2$.*

Although the new notion of conjunction introduces some syntactic redundancy with the new must transitions, it agrees with the notion given in Definition 9 when considering action-deterministic APAs.

Theorem 5. *Let N_1 and N_2 be action-deterministic APAs. We have $N_1 \oslash N_2 \preceq N_1 \wedge N_2$.*

It follows from Theorems 4 and 5 that $[\![N_1 \wedge N_2]\!] = [\![N_1 \oslash N_2]\!]$ for any two action-deterministic APAs N_1 and N_2.

Finally, just like in the case of action-deterministic APAs, non-deterministic APAs with dissimilar alphabets can be handled by first equalizing their action and atomic proposition sets using weak extension.

6 Determinism

In the previous section we have seen that the use of non-determinism changes expressiveness of APAs with respect to the known conjunction operator. In fact, non-deterministic APAs are *generally* more expressive than deterministic ones. Fig. 4 presents a non-deterministic APA, whose set of implementations cannot be specified by a single deterministic APA. States 2 and 3 have overlapping labeling constraints (so state 1 has nondeterministic behaviour). We cannot put these states on two separate a-transitions as this introduces action nondeterminism. We cannot merge them either, as their subsequent evolutions are different (and for the same reason we cannot factor $\{\alpha, \gamma\}$ to a separate state).

Nevertheless use of deterministic abstractions of non-deterministic behaviours is an interesting alternative to relying on more complex refinements and more complex operators. Below, we present a determinization algorithm that can be applied to any APA N, producing a deterministic APA $\rho(N)$, such that: $N \preceq \rho(N)$.

Our algorithm is based on subset construction and it ressembles the determinization procedure for modal transition systems described in [97].

Let $N = (S, A, L, AP, V, s_0)$ be a (consistent) APA in single valuation normal form (i.e. for all states s the set $V(s)$ is a singleton). Given a set of states $Q \subseteq S$, an action

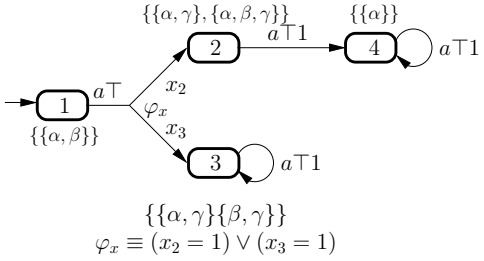


Figure 4: A (labeling) nondeterministic APA whose set of implementations cannot be obtained with a deterministic APA.

$a \in A$ and a valuation $\alpha \in AP$ we define 1-step reachability $\mathsf{Reach}(Q, a, \alpha)$ to be the maximal set of states with valuation α that can be reached with a non zero probability using a distribution π satisfying a constraint φ such that $L(q, a, \varphi) \neq \bot$ for some $q \in Q$. Formally, $\mathsf{Reach} : 2^S \times 2^A \times 2^{AP} \to 2^S$ and:

$$\mathsf{Reach}(Q, a, \alpha) = \bigcup \{s \in S \mid V(s) = \alpha \text{ and } \exists q \in Q.$$
$$\exists \varphi \in C(S). \exists \mu \in Sat(\varphi). L(q, a, \varphi) \neq \bot \text{ and } \mu(s) > 0\}$$

We lift this definition to all possible labelings as follows:

$$\mathsf{Reach}(Q, a) = \{\mathsf{Reach}(Q, a, \alpha) \mid \alpha \subseteq AP\}$$

Now define the n-step reachability as

$$\mathsf{Reach}^n(Q, a) = \mathsf{Reach}^{n-1}(Q, a) \cup \bigcup_{Q' \in \mathsf{Reach}^{n-1}(Q,a)} \mathsf{Reach}(Q', a)$$

where $\mathsf{Reach}^0(Q, a) = \{Q\}$ and denote its fixpoint as:

$$\mathsf{Reach}^*(Q, a) = \bigcup_{n=0}^{\infty} \mathsf{Reach}^n(Q, a).$$

Now, by construction, the following properties hold:

- For all $Q \subseteq S$ and $a \in A$, for all $Q', Q'' \in \mathsf{Reach}(Q, a)$, if $Q' \neq Q''$ then $Q' \cap Q'' = \emptyset$, and
- For all $Q \subseteq S$, $a \in A$ and $Q' \in \mathsf{Reach}^*(Q, a)$, there exists $\alpha \subseteq AP$ such that $\forall q' \in Q'$, we have $V(q') = \alpha$.

We will now use the notion of reachability in our determinisation construction. As already said, the algorithm works for APAs in the single valuation normal form. In [64] we show how every APA can be normalized without changing its semantics.

Definition 13. *Let $N = (S, A, L, AP, V, s_0)$ be a consistent APA in single valuation normal form. A deterministic APA for N is the APA $\rho(N) = (S', A, L', AP, V', \{s_0\})$ such that*

- $S' = \bigcup_{a \in A} \mathsf{Reach}^*(\{s_0\}, a)$
- *V' is such that $V'(Q) = \alpha$ iff $\forall q' \in Q. V(Q) = \alpha$. There always exists exactly one such α by construction*
- *L' is defined as follows: Let $Q \in S'$ and $a \in A$.*
  - *If, for all $q \in Q$, we have that $\forall \varphi \in C(S)$, $L(q, a, \varphi) = \bot$, then define $L'(Q, a, \varphi') = \bot$ for all $\varphi' \in C(S')$.*
  - *Else, define $\varphi' \in C(S')$ such that $\mu' \in Sat(\varphi')$ iff (1) $\forall Q' \notin \mathsf{Reach}(Q, a)$, we have $\mu'(Q') = 0$, and (2) there exists $q \in Q$, $\varphi \in C(S)$ and $\mu \in Sat(\varphi)$ such that $L(q, a, \varphi) \neq \bot$ and $\forall Q' \in \mathsf{Reach}(Q, a)$, $\mu'(Q') = \sum_{q' \in Q'} \mu(q')$. Then define*

$$L'(Q, a, \varphi') = \begin{cases} \top \text{ if } & \forall q \in Q, \exists \varphi \in C(S) : \\ & \quad L(q, a, \varphi) = \top \\ ? \text{ else} & \end{cases}$$

By construction, $\rho(N)$ is action- and labeling-deterministic. As expected, determinization is an abstraction. This is formalized in the following theorem.

Theorem 6. *Let N be an APA in single valuation normal form. Then $N \preceq \rho(N)$.*

7 Composition and Games

So far APAs largely rely on the composition operation defined for modal transition systems. While, this operation mimics the classical composition between transition systems, it does not allow to reason about open systems, when some transitions are not in system's control. In a series of work [84, 91], de Alfaro and Henzinger proposed an approach based on game theory for doing so. More precisely, they introduced Interface Automata, or input/output automata [125] with a game semantic. When composing two such interfaces, the algorithm identifies bad states in where one of the components can send an output that cannot be catched by the other one. Then, it computes the set of states for which there is a possibility to avoid the set of bad states. Such strategies correspond to the environments in where the components can work together.

In [82, 124], we have proposed a game semantic to modal automata by labeling may and must with input and output. In this section, we extend this setup to APAs. This extension leads to the first theory for stochastic interface automata—an optimistic extension of stochastic I/O automata [56].

7.1 Abstract Probabilistic Interfaces

We begin by introducing profiles as presented in [82].

Definition 14. *Given an alphabet of actions A, we define a profile as a function $\pi : A \to \{i, o\}$. We define $A^i = \{a \in A \mid \pi(a) = i\}$ and $A^o = \{a \in A \mid \pi(a) = o\}$, and write $\pi = (A^i, A^o)$.*

Definition 15. *Let $\pi_1 = (A_1^i, A_1^o)$ and $\pi_2 = (A_2^i, A_2^o)$ be profiles. We define the following operations:*

- *Refinement: We say that π_1 refines π_2, denoted $\pi_1 \preceq_p \pi_2$, if and only if $A_1 \supseteq A_2$ and $\pi_1(a) = \pi_2(a)$ for all $a \in A_2$*
- *Composition: If $A_1^o \cap A_2^o = \emptyset$, the composition of π_1 and π_2, denoted $\pi_1 \otimes \pi_2$, is defined as the profile $\pi_1 \otimes \pi_2 = (A^i, A^o)$ over $A_1 \cup A_2$, where $A^o = A_1^o \cup A_2^o$ and $A^i = (A_1^i \cup A_2^i) \setminus A^o$.*
- *Conjunction: If $\pi_1(a) = \pi_2(a)$ for all $a \in A_1 \cap A_2$, the conjunction of π_1 and π_2, denoted $\pi_1 \wedge \pi_2$, is defined as $A^o = A_1^o \cup A_2^o$ and $A^i = A_1^i \cup A_2^i$, where $A = A_1 \cup A_2$.*

Lemma 7. *Let $\pi_1 = (A_1^i, A_1^o)$ and $\pi_2 = (A_2^i, A_2^o)$ be profiles. If $\pi_1(a) = \pi_2(a)$ for all $a \in A_1 \cap A_2$, then*

1. *$\pi_1 \wedge \pi_2 \preceq_p \pi_1$ and $\pi_1 \wedge \pi_2 \preceq_p \pi_2$, and*
2. *whenever $\pi \preceq_p \pi_1$ and $\pi \preceq_p \pi_2$ then $\pi \preceq_p \pi_1 \wedge \pi_2$.*

We are now ready to define Abstract Probabilistic Interfaces, that are APAs whose transitions are labeled by profiles.

Definition 16. *Given an APA N with action set A and a profile $\pi : A \to \{i, o\}$, we call $\mathcal{N} = (N, \pi)$ an abstract probabilistic interface.*

Given an APA N, we use A_N and AP_N to denote the action and atomic proposition set of N, respectively.

Let I be a PA. Given a profile $\pi_I : A_I \to \{i, o\}$ and an API (N, π), if $A_I \supseteq A_N$ and $AP_I \supseteq AP_N$, we say that $\mathcal{I} = (I, \pi_I)$ satisfies (N, π), denoted $\mathcal{I} \models (N, \pi)$, if and only if $I \models N \Uparrow (A_I, AP_I)$ and $\pi_I \preceq_p \pi$. We also say that $\mathcal{I}$ is called an implementation of (N, π).

Likewise, let (N', π') be an API. If $A_N \supseteq A_{N'}$ and $AP_N \supseteq AP_{N'}$, we say that (N, π) refines an (N', π'), denoted $(N, \pi) \preceq (N', \pi')$ iff $N \preceq N' \Uparrow (A_N, AP_N)$ and $\pi \preceq_p \pi'$.

Following the presentation in [124] it is possible to express an arbitrary inteface automaton as an abstract probabilistic automaton. We refer to [123] for the translation.

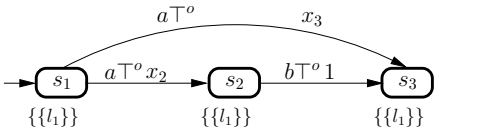


Figure 5: $\mathcal{N}_1$

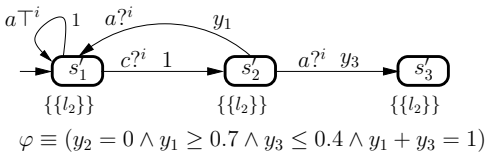


Figure 6: $\mathcal{N}_2$

7.2 Parallel Composition of Abstract Probabilistic Interfaces

We now define an optimistic composition for APIs. We start with the definition of product of two APIs.

Definition 17. *Given two APIs* $\mathcal{N}_1 = (N_1, \pi_1)$ *and* $\mathcal{N}_2 = (N_2, \pi_2)$ *with* $AP_1 \cap AP_2 = \emptyset$, *we define the product of* $\mathcal{N}_1 = (N_1, \pi_1)$ *and* $\mathcal{N}_2 = (N_2, \pi_2)$ *as* $\mathcal{N}_1 \otimes \mathcal{N}_2 = (N_1 \|_{A_1 \cap A_2} N_2, \pi_1 \otimes \pi_2)$.

For two APIs N_1 and N_2 define the set of bad states $\mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2}$ as the set of pairs $(s_1, s_2) \in S_1 \times S_2$ satisfying one of the two following conditions:

1. There exists $a \in A_1^o \cap A_2^i$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, a, \varphi_1) \geq ?$ and for all $\varphi_2 \in C(S_2)$, $L_2(s_2, a, \varphi_2) \neq \top$, or
2. There exists $a \in A_2^o \cap A_1^i$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, a, \varphi_2) \geq ?$ and for all $\varphi_1 \in C(S_1)$, $L_1(s_1, a, \varphi_1) \neq \top$.

Basically, a state of the product is a bad if one of the operands can send an action that the other operand may avoid to catch.

Example 6. *Consider the APIs* $\mathcal{N}_1$ *and* $\mathcal{N}_2$ *given in Fig. 5 and Fig. 6. Their profiles are specified by attaching o and i letters to transition labels. The API* $\mathcal{N}_1 \otimes \mathcal{N}_2$ *is given in Fig. 7. Observe that* $(s_1, s'_2) \in \mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2}$. *Indeed, the action a is an output action of* $\mathcal{N}_1$ *and an input action of* $\mathcal{N}_2$. *However, while there is a may-transition from* s_1 *on a, there is no must-transition on a from* s'_2.

We now propose an algorithm that computes the set of states from where there is a way to reach the set of bad states. Given an API $\mathcal{N} = (N, \pi)$ on state set S and action set A, the function pre of a set $S' \subseteq S$, $\mathsf{pre}^1(S')$ is defined as

$$\mathsf{pre}^1(S') = \{s \in S \mid \exists a \in A^o. \exists \varphi \in C(S). \exists \mu \in Sat(\varphi). L(s, a, \varphi) \geq ? \wedge \mu(S') > 0\}$$

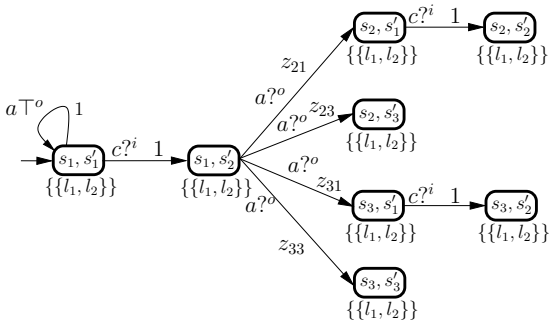


Figure 7: $\mathcal{N}_1 \otimes \mathcal{N}_2$

We say that $\mathsf{pre}^0(S') = S'$, for $k \geq 0$, $\mathsf{pre}^{k+1}(S') = \mathsf{pre}^1(\mathsf{pre}^k(S'))$, and $\mathsf{pre}(S') = \bigcup_{k \geq 0} \mathsf{pre}^k(S')$.

Definition 18. *We say that APIs $\mathcal{N}_1 = (N_1, \pi_1)$ and $\mathcal{N}_2 = (N_2, \pi_2)$ are compatible, if $(s_0^1, s_0^2) \notin \mathsf{pre}(\mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2})$.*

After computing $\mathcal{N}_1 \otimes \mathcal{N}_2$ and $\mathsf{pre}(\mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2})$, the product is *relaxed*. For each state $s \in S$ and each action $a \in A$ perform the following: if it is possible to reach one of the states in $\mathsf{pre}(\mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2})$ with non-zero probability after issuing a then redefine s to have only a may-transition on a, with the constraint containing only a distribution giving probability 1 to a fresh state s_{may} not in S that allows everything but does not require anything.

Definition 19. *Given two compatible APIs $\mathcal{N}_1 = (N_1, \pi_1)$ and $\mathcal{N}_2 = (N_2, \pi_2)$, we define the composition of $\mathcal{N}_1$ and $\mathcal{N}_2$, denoted as $\mathcal{N}_1 \| \mathcal{N}_2$, as the API obtained by substituting L and V in $\mathcal{N}_1 \otimes \mathcal{N}_2$ by L', a copy of L that is manipulated in the following way, and V', an extension of V: For all $(s_1, s_2) \in S_1 \times S_2$ and for all $a \in A$, if $(s_1, s_2) \notin \mathsf{pre}(\mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2})$ and there exists $\varphi \in C(S)$ and $\mu \in Sat(\varphi)$ such that*

$$L((s_1, s_2), a, \varphi) \geq ? \wedge \mu(\mathsf{pre}(\mathsf{bad}_{\mathcal{N}_1 \otimes \mathcal{N}_2})) > 0$$

then let $L'((s_1, s_2), a, \varphi) = \bot$ and $L'((s_1, s_2), a, \varphi') = L((s_1, s_2), a, \varphi)$, where $Sat(\varphi') = \{\mu'\}$ and $\mu'(s_{may}) = 1$.

The new state s_{may}, not in $S_1 \times S_2$, is defined as, for all $a \in A$, $L'(s_{may}, a, \varphi'') = ?$, where $\varphi'' \in C(S)$ is such that $Sat(\varphi'') = \{\mu''\}$ and $\mu''(s_{may}) = 1$. The function V' is defined as $V'(s_1, s_2) = V(s_1, s_2)$ for all $(s_1, s_2) \in S_1 \times S_2$ and $V'(s_{may}) = 2^{AP}$.

Example 7. *Returning to Example 6, the parallel composition of $\mathcal{N}_1$ and $\mathcal{N}_2$ is obtained as the API in Fig. 8. The profile of the composition is $\pi_1 \otimes \pi_2 = [a \mapsto o, b \mapsto o, c \mapsto i]$.*

Since (s_1, s_2') is a bad state it becomes unreachable (and thus most of other state pairs become unreachable). Instead a transition to the universal state s_{may} is inserted, modeling the fact that after receiving c the system becomes unpredictable.

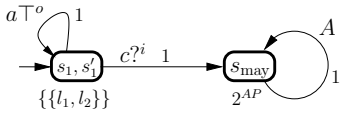


Figure 8: $\mathcal{N}_1 \| \mathcal{N}_2$

Our composition for APIs satisfies classical theorems of independent implementability.

Lemma 8. *Given two compatible APIs $\mathcal{N}_1$ and $\mathcal{N}_2$, it holds that $\mathcal{N}_1 \otimes \mathcal{N}_2 \preceq \mathcal{N}_1 \| \mathcal{N}_2$.*

Theorem 9. *Given two compatible APIs $\mathcal{N}_1$ and $\mathcal{N}_2$, and two implementations $\mathcal{I}_1$ and $\mathcal{I}_2$, such that $\mathcal{I}_1 \models \mathcal{N}_1$ and $\mathcal{I}_2 \models \mathcal{N}_2$, then $\mathcal{I}_1 \otimes \mathcal{I}_2 \models \mathcal{N}_1 \| \mathcal{N}_2$.*

8 Implementation

Some of the operations introduced in this paper have been implemented in a new tool, written in C# 4.0, called APAC.[1] To the best of our knowledge, this is the first implementation effort for stochastic interfaces. Presently the tool relies on the Z3 solver [112] for solving constraints. The tool implements the following operations: weak refinement checking, weak weak refinement checking, determinism checking, pruning (β^*), alphabet extensions, and conjunction.

Example 8. *We present the input format of the tool in an example. We will be checking weak refinement between two APAs. The following code example will result in the definition of two models N_1 and N_2. The last line specifies that weak refinement will be checked. Notice, that we require states to be named with the natural numbers where 1 is the initial state.*

```
Name: N1;
A:(a,b);
AP:(l,m,n,o);
state 1:((l)): a? -> x[1] = 0.0 && x[2] + x[3] >= 7/10 && x[3] + x[4] >=
    2/10;
state 2:((m)): b? -> x[1] = 0.0 && x[2] = 0.0 && (x[3] = 1.0 || x[4] =
   1.0);
state 3:((n)): b? -> x[3] = 1.0;
state 4:((o)): b? -> x[4] = 1.0;

Name: N2;
A:(a,b);
AP:(l,m,n,o);
state 1:((l)): a? -> x[1] = 0.0 && x[2]+x[3] >= 7/10 && x[4] + x[5] >=
   2/10;
```

[1] The tool can be found on `www.cs.aau.dk/~mikkelp/apac`

```
state 2:((m)): b? -> x[3] <= 1.0 && x[4] <= 1.0 && x[5] <= 1.0 && x[1] =
     0.0 && x[2] = 0.0;
state 3:((n)): b? -> x[4] = 1.0;
state 4:((n)): b? -> x[3] = 1.0;
state 5:((o)): b? -> x[5] = 1.0;

check: N1 wref N2;
```

It takes 179 milliseconds on a typical laptop before APAC reports that $\{(1,1),(2,2),(3,3),(3,4),(4,5)\}$ *is a weak refinement relation.*

At the moment APAC does not support parallel composition. This is because its definition requires use of multiplication, which is not supported by Z3. The situation could have been the same for refinement, but we have been able to use a different encoding. The idea is to let the correspondence functions give the actual value redistributed, and not the proportions. Still, this trick cannot be used for strong refinement, as defined in [64]. At the moment we do not know, whether strong refinement can be checked relying solely on solving linear arithmetic constraints.

We discuss the weak refinement in somewhat more details. The algorithm is implemented as a coinductive fixpoint iteration. Starting from the full relation, violating pairs are removed until a fixpoint is reached. Given a pair of states $(s,s') \in \mathcal{R}$ and constraints φ and φ', the pseudo-formula, Eq. (13), is passed to Z3. We invoke quantifier elimination, and since all variables are quantified, quantifier elimination will evaluate the formula to true or false.

$$
\begin{aligned}
\forall x : \varphi(x) \Rightarrow \exists \delta : S \to (S' \to [0,1]) : \\
\varphi'\left(t \mapsto \sum_{s \in S} \delta(s)(t)\right) \wedge \\
\forall s \in S : x_s = \sum_{s' \in S'} \delta(s)(s') \wedge \\
\forall (s,s') : \left[(s,s') \notin \mathcal{R} \vee V(s) \not\subseteq V'(s')\right] \\
\qquad \Rightarrow \delta(s)(s') = 0 \wedge \\
\forall (s,s') : \left[(s,s') \in \mathcal{R} \wedge V(s) \subseteq V'(s')\right] \\
\qquad \Rightarrow 0 \leq \delta(s)(s') \leq 1
\end{aligned}
\tag{13}
$$

Notice that, if a pair of states has conflicting labeling, we set the correspondence function to 0 for this pair. The tool can also synthesize a witness in case the refinement does not hold.

In order to evaluate the performance of the tool, we generate "random" APAs and measure the time for performing operations on these. Given a number of states and whether or not we are interested in simple or more elaborate constraints, we generate an APA with an action alphabet A and an atomic proposition alphabet AP on 5–10 members each, state valuations consisting of up to 0–4 members of 2^{AP}, 1–3 outgoing transitions for each state on random action and modality, and a random choice between

APA 1		APA 2		
states	simple	states	simple	time
10	yes	10	yes	10/20/72 ms
10	no	10	no	10/41/1121 ms
10	no	10	yes	20/1046/? ms
10	yes	10	no	18/94/4079 ms
15	yes	15	yes	125/140/? ms

Table 9: Weak refinement

constraint designs for each transition. Given a state i there are three simple constraints and three more eleborate constraints:

- simple:
  - $x_{i+1} \geq 7/10 \wedge x_{i+2} \leq 3/10$,
  - $x_{i+1} = 7/10 \wedge x_{i+2} = 3/10$, and
  - $x_{i+1} = 1.0$
- more elaborate:
  - $x_{i+1} \geq 3/10 \wedge x_{i+1} \leq 4/10$,
  - true, and
  - $x_{i+1} = 1.0 \vee (x_{i+1} \geq 7/10 \wedge x_{i+2} \leq 3/10)$

The tests, that are summarized in Table 9, are performed on an x64 Intel Core 2 Duo 2.2 GHz with 4 GB RAM running Windows 7, using version 2.16 of the Z3 API. The first line of the table gives execution times for three random input files (three times are reported, as the experiment was repeated three times, with different randomly generated instances). In each input file, weak refinement is checked on two random APAs on each 10 states with simple constraints.

This procedure is repeated for each line in the table. A question mark (?) means that the specific random input file does not stop executing within 5 minutes.

The above results are still preliminary and we hope to reduce the computation time by adapting classical heuristics for fixed-point computations [74].

9 Conclusion

In [64], we have introduced the first complete specification theory for PAs with a comparison operator and both logical and structural composition. In this paper, we have strengthened those results by extending the power of the operators as well as the expressiveness of the model. The results have been implemented in APAC, a prototype tool that has been evaluated on several case studies.

There are many directions for future research. First, one should pursue the development of APAC by adding the composition operators. Heuristics should also be implemented. Among them, one naturally thinks of the work by Henzinger et al.[74] that could be adapted to reduce the number of steps in the fixed-point algorithm for refinement. Another suggestion would be to adapt bisimulation quotient [126] in order to minimize the size of the APAs.

Another direction is to develop a generalized model checking procedure for APAs. We postulate that this could be done by extending results obtained for Hennessy-Milner logic and modal automata [42]. Finally, we are also considering to mix the results on APAs with those we obtained on timed interfaces. This would lead to the first specification theory for timed systems [106] with a stochastic semantics.

Paper F – APAC: a tool for reasoning about Abstract Probabilistic Automata

Benoit Delahaye
INRIA/IRISA, Rennes

Kim G. Larsen
Aalborg University

Axel Legay
INRIA/IRISA, Rennes

Mikkel L. Pedersen
Aalborg University

Andrzej Wasowski
IT University, Copenhagen

1 Abstract

We recently introduced Abstract Probabilistic Automata (APA), a new powerful abstraction formalism for probabilistic automata. Our theory is equipped with a series of aggressive abstraction techniques for state-space reduction as well as a specification theory for both logical and structural comparisons. This paper reports on the implementation of the approach in the Abstract Probabilistic Automata Checker toolset.

2 Context

Probabilistic Automata (PA) [6] is a formalism for modeling systems that exhibit stochastic and non-deterministic behaviour, e.g., randomized distributed algorithms and fault tolerant systems. In each state, a PA resolves a non-deterministic choice and then moves to the successor state according to some probability distribution. Recently [64, 65], we proposed Abstract Probabilistic Automata (APA), a new powerful abstraction formalism for PAs equipped with (1) a series of aggressive abstraction techniques for state-space reduction, and (2) a specification theory for component-based design in the spirit of [84]. This short paper reports on the Abstract Probabilistic Au-

tomata Checker toolset (APAC), an implementation of the APAs theory (and hence of the first interface theory for stochastic systems), based on the SMT-solver Z3 [112]. The tool, tutorials and a manual can be found at `http://www.cs.aau.dk/~mikkelp/apac`.

The APA model. Syntactically, an APA is a PA whose transitions are equipped with may and must modalities [42], and whose probability distributions are replaced by constraints like in Constraint Markov Chains (CMC) [63]. Semantically, an APA represents a possibly infinite set of PAs that are its implementations. Each state is also equipped with a set of atomic propositions used to define additional hypotheses on the implementations. The must modality requires that the transition has to be present in any implementation, while the may modality permits its absence. In addition, any distribution associated to the transition must satisfy the constraint specified by the APA. Consider APA N_2 of Fig. 1b. Three transitions leave the state s'_1: an a-must-transition to constraint φ', an a-may-transition to a constraint assigning probability 1 to s'_5, and a b-may-transition going with probability 1 to s'_1. Any implementation of N_2 has an a-transition targeting a distribution satisfying φ'; the other transitions are optional.

Abstraction. We propose two notions of abstraction. The first one is classical and aims at reducing the state-space of the system by lumping equivalence classes. The second one is used to abstract a constraint φ by the smallest intervals in which all satisfying distributions can be embedded.

Specification Theory. The APA model also serves as a specification theory for stochastic systems. This is used to decompose the design and hence reduce its complexity. We propose a structural composition that allows to combine components and a logical composition that allows to combine requirements (take intersection of sets of implementations). These operations unite those defined on modal automata [42] and CMCs [63]. For example, in order to structurally compose two APAs, we combine transitions labeled by the same letter. The combination of a may with a must or a may transition transition leads to a may transition. Constraints are combined in a product-like manner. Given constraints $\varphi_1 \in C(S_1)$ and $\varphi_2 \in C(S_2)$, their product $\varphi_1\varphi_2$ is defined such that for all distribution μ satisfying $\varphi_1\varphi_2$, written $\mu \in Sat(\varphi_1\varphi_2)$, there exists $\mu_1 \in Sat(\varphi_1)$ and $\mu_2 \in Sat(\varphi_2)$ such that $\mu(s_1, s_2) = \mu_1(s_1)\mu_2(s_2)$ for all $(s_1, s_2) \in S_1 \times S_2$.

Refinement. Refinement compares APAs and hence also sets of implementations. Intuitively, if N_1 refines N_2, then any must (resp. may) of N_2 (resp. N_1) should be matched in N_1 (resp. N_2) in an alternating simulation manner. Moreover, the matching has to agree on the constraints as illustrated hereafter. Consider the two APAs in Fig. 1 with state space S and S', respectively. $\mathcal{R} = \{(s_1, s'_1), (s_2, s'_2), (s_3, s'_3), (s_3, s'_4), (s_4, s'_5)\}$ is a refinement relation. Indeed, the a-must-transition from s'_1 is matched by a must-transition in s_1, and the b-must-transition from s_1 is matched in s'_1, and φ and φ' agree. Indeed, for all distributions μ that satisfy φ, the probability mass given to successor states by μ can be redistributed to equal a distribution μ' satisfying φ'. Let $\delta : S \to (S' \to [0, 1])$ be given as $(s_1, s'_1) \mapsto 1$, $(s_2, s'_2) \mapsto 1$, $(s_3, s'_3) \mapsto \gamma$, $(s_3, s'_4) \mapsto 1-\gamma$, $(s_4, s'_5) \mapsto 1$, and 0 else, where $\gamma = \frac{0.7-\mu(s_2)}{\mu(s_3)}$, if $\mu(s_2) \leq 0.7$, and $\gamma = \frac{0.8-\mu(s_2)}{\mu(s_3)}$ else. For

all distributions μ that satisfies φ, $\mu\delta$ will satisfy φ', and for all pairs (s, s') such that $\delta(s)(s') > 0$, $s \mathcal{R} s'$. Details can be found in [64, 65].

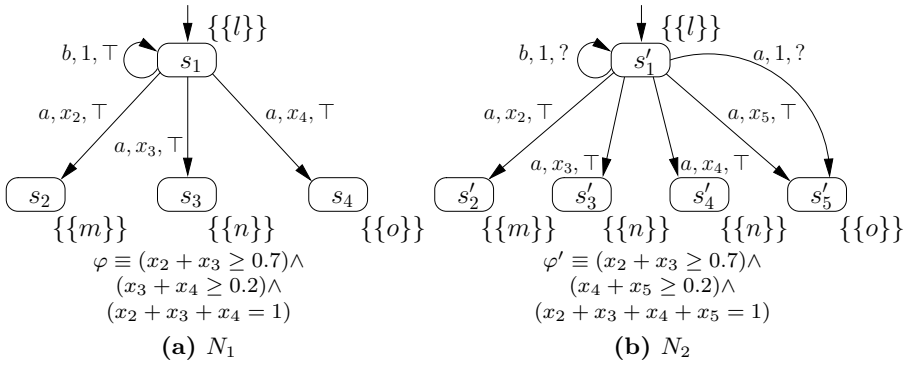


Figure 1: APAs N_1 and N_2.

3 The APAC Tool

Input Language. APAC provides a simple intuitive textual language for specifying APAs and operations on them. The language follows the graphical description of APAs. For example, the two APAs of Fig. 1 can be described as shown in Fig. 2. Each state is declared using the `state` keyword followed by a non-zero integer. Transitions are declared by their modalities (! for must and ? for may) followed by the keyword $\rightarrow$ and the constraint on the distribution on the successor states. For example, $x[2] + x[3] \geq 0.7$ imposes that the probability mass assigned to states 2 and 3 is greater than 0.7.

APAC, itself implemented in C#, parses the input language and builds internal representations for the corresponding APAs. Then, operations are applied to create new APAs. All the operations are reduced to suitable constraint manipulations in Z3 [112], an efficient SMT solver supporting quantifier elimination. For instance, in order to check refinement we perform quantifier elimination in the formula $\forall \mu \in Sat(\varphi), \exists \delta, \exists \mu' \in Sat(\varphi') : \mu\delta = \mu'$. As all variables are quantified, the procedure will evaluate the formula to true or false. If refinement does not hold, then APAC can generate a counter example.

Due to limitations of Z3, APAC only handles linear constraints. Fortunately, linearity of constraints is known to be preserved by all the operations in our theory, except structural composition. We solve this problem by using a linear abstraction of the constraints. Given constraints $\varphi_1 \in C(S_1)$ and $\varphi_2 \in C(S_2)$, their combination φ is defined such that for all $\mu \in Sat(\varphi)$, $\sum_{s_1 \in S_1} \mu(s_1, s_2) \in Sat(\varphi_2)$ and $\sum_{s_2 \in S_2} \mu(s_1, s_2) \in Sat(\varphi_1)$. Also in other areas, such as control theory, non-linear systems have to be abstracted by linear ones for efficiency reasons.

APAC supports generalized model checking of a disjunction-free extension of Hennessy-Milner logic [127] over APAs. For example, the formula $[a]_{\geq 0.2}\{\{n\}\}$ specifies that

```
INPUT:
Name: N1;
A:(a,b);
AP:(l,m,n,o);
state 1:((l)):
a!->x[1]=0.0 && x[2]+x[3]>=7/10 && x[3]+x[4]>=2/10;
b!->x[1]=1.0;
state 2:((m)):
state 3:((n)):
state 4:((o)):

Name: N2;
A:(a,b);
AP:(l,m,n,o);
state 1:((l)):
a!->x[1]=0.0 && x[2]+x[3]>=7/10 && x[4]+x[5]>=2/10;
a?->x[5]=1.0;
b?->x[1]=1.0;
state 2:((m)):
state 3:((n)):
state 4:((n)):
state 5:((o)):

check: N1 wref N2;
```

Figure 2: Invoking refinement checking

Table 10: State abstraction

states	abstract states	time
500	5	9509 ms
500	10	16213 ms
500	50	62727 ms
500	100	96399 ms

for all a-may-transitions leading to constraint φ, it holds that all satisfying distributions of φ give probability of at least 0.2 to states with valuations being subset of $\{\{n\}\}$. Obviously N_1 does not satisfy this formula. The full definition and input grammar of the logic can be found at `http://www.cs.aau.dk/~mikkelp/apac`. The logic is sound and complete i.e. an APA N satisfies a formula φ if and only if all implementations of N satisfy φ.

4 Results and conclusion

APAC is clearly a research tool, still undergoing heavy development. While not yet mature enough to handle industrial case studies, it already is the first ever implementation of a compositional specification theory for stochastic systems. Already now APAC is able to decide refinement of large-size case studies. Not surprisingly, the running time of quantifier elimination increases using an increasing number of quantified variables. In Table 10, we see that time increases linearly with the precision of the state abstraction. More details are available on the APAC website.

In the future we intend to improve the efficiency of APAC by implementing heuristics such as bisimulation reduction for APAs. We also plan to implement a graphical user interface. Here the main challenge is identifying a simple and easy to manipulate representation for transitions encompassing multiple arrows related with a probability distribution constraint.

APAC is a part of a broader effort to develop and apply specification theories

to industrial problems [128]. Recently, we have achieved success with the ECDAR toolset [129] and modeling of real time systems. It is of interest to merge ECDAR and APAC, but this requires developing a new specification theory first.

References

[1] T. A. Henzinger and J. Sifakis, "The embedded systems design challenge," in *Formal Methods (FM)*, 2006.

[2] G. Le Lann, "An analysis of the Ariane 5 flight 501 failure - a system engineering perspective," in *Engineering of Computer-Based Systems (ECBS)*, 1997.

[3] M. Fruth, "Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol," in *International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA)*, 2006.

[4] M. Stoelinga and F. Vaandrager, "Root contention in IEEE 1394," in *AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems (ARTS)*, 1999.

[5] A. Markov, "Extension of the law of large numbers to dependent variables," in *Selected Works*, 1951.

[6] R. Segala and N. Lynch, "Probabilistic simulations for probabilistic processes," in *Concurrency Theory (CONCUR)*, 1994.

[7] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logic of Programs, Workshop*, 1982.

[8] "UPPAAL," http://www.uppaal.org, webpage.

[9] K. L. McMillan, "Symbolic model checking: An approach to the state space explosion problem," Ph.D. dissertation, Carnegie Mellon University, 1992.

[10] G. J. Holzmann, "The model checker spin," *IEEE Transactions on Software Engineering (TSE)*, 1997.

[11] M. Z. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic symbolic model checking with PRISM: a hybrid approach," *Software Tools for Technology Transfer (STTT)*, 2004.

[12] M. Huth and M. Kwiatkowska, "Quantitative analysis and model checking," in *Logic in Computer Science (LICS)*, 1997.

[13] A. Pnueli, "The temporal logic of programs," in *Symposium on Foundations of Computer Science (SFCS)*, 1977.

[14] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1986.

[15] C. Baier and J.-P. Katoen, *Principles of Model Checking.* The MIT Press, 2008.

[16] K. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Information and Computation (I&C)*, 1991.

[17] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton, "Verifying continuous time markov chains," in *Computer Aided Verification (CAV)*, 1996.

[18] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing (BCS-FACS)*, 1994.

[19] R. M. Keller, "Formal verification of parallel programs," *Communications of the ACM (CACM)*, 1976.

[20] J. C. Baeten, "A brief history of process algebra," *Theoretical Computer Science (TCS)*, 2005.

[21] D. Park, "Concurrency and automata on infinite sequences," *Theoretical Computer Science (TCS)*, 1981.

[22] Robin and Milner, "Calculi for synchrony and asynchrony," *Theoretical Computer Science (TCS)*, 1983.

[23] J. C. M. Baeten, T. Basten, and M. A. Reniers, *Process Algebra: Equational Theories of Communicating Processes.* Cambridge University Press, 2009.

[24] R. Milner, *Communication and concurrency.* Prentice-Hall, 1989.

[25] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM (CACM)*, 1978.

[26] J. Bergstra and J. Klop, "Process algebra for synchronous communication," *Information and Control*, 1984.

[27] F. Moller and P. Stevens, "Edinburgh Concurrency Workbench user manual (version 7.1)," http://homepages.inf.ed.ac.uk/perdita/cwb/.

[28] H. Hansson and B. Jonsson, "A calculus for communicating systems with time and probabilities," in *Real-Time Systems Symposium (RTSS)*, 1990.

[29] S. Andova, "Process algebra with probabilistic choice," in *AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems (ARTS)*, 1999.

[30] B. Jonsson, K. G. Larsen, and W. Yi, "Probabilistic extensions of process algebras," in *Handbook of Process Algebra*, 2001.

[31] Y. Deng and W. Du, "Probabilistic barbed congruence," *Electronic Notes in Theoretical Computer Science (ENTCS)*, 2007.

[32] C.-C. Jou and S. Smolka, "Equivalences, congruences, and complete axiomatizations for probabilistic processes," in *Concurrency Theory (CONCUR)*, 1990.

[33] R. Mardare and C. Priami, "Decidable extensions of Hennessy-Milner Logic," in *Formal Methods for Networked and Distributed Systems (FORTE)*, 2006.

[34] L. Cardelli, K. G. Larsen, and R. Mardare, "Modular markovian logic," in *International Colloquium on Automata, Languages and Programming (ICALP)*, 2011.

[35] K. G. Larsen and B. Thomsen, "A modal process logic," in *Logic in Computer Science (LICS)*, 1988.

[36] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, and R. Passerone, "Why are modalities good for interface theories?" in *Application of Concurrency to System Design (ACSD)*, 2009.

[37] G. Boudol and K. G. Larsen, "Graphical versus logical specifications," *Theoretical Computer Science (TCS)*, 1992.

[38] K. G. Larsen, L. Juhl, and J. Srba, "Modal transition systems with weight intervals," in *Journal of Logic and Programming (JLAP)*, 2011.

[39] S. S. Bauer, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski, "A modal specification theory for components with data," in *Formal Aspects of Component Software (FACS)*, 2011, to appear in LNCS.

[40] M. H. Møller, N. Benes, J. Kretinsky, K. G. Larsen, and J. Srba, "Parametric modal transition systems," in *Automated Technology for Cerification and Analysis (ATVA)*, 2011.

[41] J.-B. Raclet, "Residual for component specifications," *Electronic Notes in Theoretical Computer Science (ENTCS)*, 2008.

[42] K. G. Larsen, "Modal specifications," in *Automatic Verification Methods for Finite State Systems (AVMS)*, 1989.

[43] N. Benes, J. Kretínský, K. G. Larsen, and J. Srba, "Checking thorough refinement on modal transition systems is EXPTIME-complete," in *International Colloquium on Theoretical Aspects of Computing (ICTAC)*, 2009.

[44] B. Delahaye, K. G. Larsen, A. Legay, and A. Wasowski, "On greatest lower bound of modal transition systems," INRIA/IRISA, ITU, AAU, Tech. Rep., 2010. [Online]: http://www.irisa.fr/s4/people/benoit.delahaye/rapports/modalIPL.pdf

[45] N. Bertrand, S. Pinchinat, and J.-B. Raclet, "Refinement and consistency of timed modal specifications," in *Language and Automata Theory and Applications (LATA)*, 2009.

[46] B. Jonsson and K. G. Larsen, "Specification and refinement of probabilistic processes," in *Logic in Computer Science (LICS)*, 1991.

[47] J. Katoen, D. Klink, M. Leucker, and V. Wolf, "Three-valued abstraction for continuous-time Markov chains," in *Computer Aided Verification (CAV)*, 2007.

[48] H. Fecher, M. Leucker, and V. Wolf, "Don't know in probabilistic systems," in *Model Checking Software*, 2006.

[49] J.-P. Katoen, D. Klink, and M. R. Neuhäußer, "Compositional abstraction for stochastic systems," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2009.

[50] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wasowski, "Modal and mixed specifications: key decision problems and their complexities," *Mathematical Structures in Computer Science (MSCS)*, 2010.

[51] R. Lanotte, A. Maggiolo-Schettini, and A. Troina, "Parametric probabilistic transition systems for system design and analysis," *Formal Aspects of Computing*, 2007.

[52] R. Segala and N. A. Lynch, "Probabilistic simulations for probabilistic processes," *Nordic Journal of Computing (NJC)*, 1995.

[53] R. Segala, "Modeling and verification of randomized distributed real-time systems," Ph.D. dissertation, Massachusetts Institute of Technology, 1995.

[54] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[55] M. Stoelinga, "An introduction to probabilistic automata," *Bulletin of The European Association for Theoretical Computer Science (BEATCS)*, 2002.

[56] N. Lynch, I. Saias, and R. Segala, "Proving time bounds for randomized distributed algorithms," in *Principles of Distributed Computing (PODC)*, 1994.

[57] L. Zhang and H. Hermanns, "From concurrency models to numbers - performance and dependability," in *NATO Science for Peace and Security Series - D: Information and Communication Security*. IOS Press, 2009.

[58] A. Dolzmann and T. Sturm, "REDLOG: computer algebra meets computer logic," *SIGSAM Bull.*, 1997.

[59] C. W. Brown, "QEPCAD B: a program for computing with semi-algebraic sets using CADs," *SIGSAM Bull.*, 2003.

[60] S. Sankaranarayanan, H. Sipma, and Z. Manna, "Constraint-based linear-relations analysis," in *Static Analysis Symposium (SAS)*, 2004.

[61] B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski, "Decision problems for Interval Markov Chains," in *Language and Automata Theory and Applications (LATA)*, 2011.

[62] B. Caillaud, B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski, "Constraint Markov Chains," *Theoretical Computer Science (TCS)*, 2011.

[63] ——, "Compositional design methodology with Constraint Markov Chains," in *Quantitative Evaluation of Systems (QEST)*, 2010.

[64] B. Delahaye, J.-P. Katoen, K. G. Larsen, A. Legay, M. L. Pedersen, F. Sher, and A. Wasowski, "Abstract Probabilistic Automata," in *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, 2011.

[65] ——, "New Results on Abstract Probabilistic Automata," in *Application of Concurrency to System Design (ACSD)*, 2011.

[66] B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski, "APAC: a tool for reasoning about Abstract Probabilistic Automata," in *Quantitative Evaluation of Systems (QEST)*, 2011.

[67] ——, "New results for Constraint Markov Chains," *Performance Evaluation (PEVA)*, 2011.

[68] N. López and M. Núñez, "An overview of probabilistic process algebras and their equivalences," in *Validation of Stochastic Systems (VSS)*, 2004.

[69] H. J. Hartfield, *Markov Set-Chains*. Springer, 1998.

[70] A. Abate, A. D'Innocenzo, M. D. D. Benedetto, and S. S. Sastry, "Markov set-chains as abstractions of stochastic hybrid systems," in *Hybrid Systems: Computation and Control (HSCC)*, 2008.

[71] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1994.

[72] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *Journal of the ACM (JACM)*, 2003.

[73] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wasowski, "20 years of modal and mixed specifications," *Bulletin of The European Association for Theoretical Computer Science (BEATCS)*, 2008.

[74] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, "Computing simulations on finite and infinite graphs," in *Foundations of Computer Science (FOCS)*, 1995.

[75] D. Dams, "Abstract interpretation and partition refinement for model checking," Ph.D. dissertation, Eindhoven University of Technology, 1996.

[76] K. Sen, M. Viswanathan, and G. Agha, "Model-checking Markov chains in the presence of uncertainties," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2006.

[77] K. Chatterjee, K. Sen, and T. A. Henzinger, "Model-checking omega-regular properties of interval Markov chains," in *Foundations of Software Science and Computation Structures (FoSSaCS)*, 2008.

[78] S. Haddad and N. Pekergin, "Using stochastic comparison for efficient model checking of uncertain Markov chains," in *Quantitative Evaluation of SysTems (QEST)*, 2009.

[79] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement," in *Computer Aided Verification (CAV)*, 2000.

[80] M. O. Rabin, "Probabilistic automata," *Information and Control*, 1963.

[81] K. G. Larsen, U. Nyman, and A. Wasowski, "On modal refinement and consistency," in *Concurrency Theory (CONCUR)*, 2007.

[82] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone, "Modal interfaces: unifying interface automata and modal specifications," in *Embedded Software (EMSOFT)*, 2009.

[83] K. G. Larsen, U. Nyman, and A. Wasowski, "Modal I/O automata for interface and product line theories," in *European Symposium on Programming (ESOP)*, 2007.

[84] L. de Alfaro and T. A. Henzinger, "Interface automata," in *Foundations of Software Engineering (FSE)*, 2001.

[85] L. Doyen, T. A. Henzinger, B. Jobstmann, and T. Petrov, "Interface theories with component reuse," in *Embedded Software (EMSOFT)*, 2008.

[86] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and F. Y. C. Mang, "Synchronous and bidirectional component interfaces," in *Computer Aided Verification (CAV)*, 2002.

[87] L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, P. Roy, and M. Sorea, "Sociable interfaces," in *Frontiers of Combining Systems (FroCos)*, 2005.

[88] B. T. Adler, L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, V. Raman, and P. Roy, "Ticc: A tool for interface compatibility and composition," in *Computer Aided Verification (CAV)*, 2006.

[89] T. Brazdil, V. Forejt, J. Kretinsky, and A. Kucera, "The satisfiability problem for Probabilistic CTL," in *Logic in Computer Science (LICS)*, 2008.

[90] A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski, "Methodologies for specification of real-time systems using Timed I/O Automata," in *Formal Methods for Components and Objects (FMCO)*, 2009.

[91] L. de Alfaro and T. A. Henzinger, "Interface-based design," in *Engineering Theories of Software-intensive Systems*, 2005.

[92] H. Hermanns, *Interactive Markov Chains and the Quest for Quantified Quality.* Springer, 2002.

[93] C. W. Brown, "Simple CAD construction and its applications," *Journal of Symbolic Compututation (JSC)*, 2001.

[94] C. W. Brown and J. H. Davenport, "The complexity of quantifier elimination and cylindrical algeraic decomposition," in *Symbolic and Algebraic Computation (SSAC)*, 2007.

[95] H. Yanami and H. Anai, "SyNRAC: a Maple toolbox for solving real algebraic constraints," *ACM Communications in Computer Algebra*, 2007.

[96] S. Basu, "New results on quantifier elimination over real closed fields and applications to constraint databases," *Journal of the ACM (JACM)*, 1999.

[97] N. Beneš, J. Křetínský, K. Larsen, and J. Srba, "On determinism in modal transition systems," *Theoretical Computer Science (TCS)*, 2009.

[98] F. Ciesinski and M. Größer, "On probabilistic computation tree logic," in *Validation of Stochastic Systems (VSS)*, 2004.

[99] H. Hermanns, U. Herzog, and J. Katoen, "Process algebra for performance evaluation," *Theoretical Computer Science (TCS)*, 2002.

[100] J. Hillston, *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[101] H. Hermanns, B. Wachter, and L. Zhang, "Probabilistic CEGAR," in *Computer Aided Verification (CAV)*, 2008.

[102] E. Altman, *Constrained Markov Decision Processes.* Chapman & Hall/CRC, 1999.

[103] K. G. Larsen and A. Skou, "Compositional verification of probabilistic processes," in *Concurrency Theory (CONCUR)*, 1992.

[104] J.-B. Raclet, "Quotient de spécifications pour la réutilisation de composants," Ph.D. dissertation, Université de Rennes I, 2007.

[105] P. Bhaduri, "Synthesis of interface automata," in *Automated Technology for Cerification and Analysis (ATVA)*, 2005.

[106] A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski, "Timed I/O Automata : A complete specification theory for real-time systems," in *Hybrid Systems: Computation and Control (HSCC)*, 2010.

[107] N. Bertrand, A. Legay, S. Pinchinat, and J.-B. Raclet, "A compositional approach on modal specifications for timed systems," in *International Conference on Formal Engineering Methods (ICFEM)*, 2009.

[108] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang, "Symbolic model checking for probabilistic timed automata," *Information and Computation (I& C)*, 2007.

[109] B. Delahaye, B. Caillaud, and A. Legay, "Probabilistic contracts: A compositional reasoning methodology for the design of stochastic systems," in *Application of Concurrency to System Design (ACSD)*, 2010.

[110] L. de Alfaro, T. A. Henzinger, and R. Jhala, "Compositional methods for probabilistic systems," in *Concurrency Theory (CONCUR)*, 2001.

[111] T. Bourke, A. David, K. G. Larsen, A. Legay, D. Lime, U. Nyman, and A. Wasowski, "New results on timed specifications," in *Workshop on Algebraic Development Techniques (WADT)*, 2010.

[112] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2008.

[113] A. Arnold, "MEC: a system for constructing and analysing transition systems," in *Automatic verification methods for finite state systems (AVMFSS)*, 1990.

[114] "Maple," http://www.maplesoft.com/products/Maple/index.aspx, webpage.

[115] J.-P. Katoen, T. Kemna, I. S. Zapreev, and D. N. Jansen, "Bisimulation minimisation mostly speeds up probabilistic model checking," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2007.

[116] R. Segala, "Probability and nondeterminism in operational models of concurrency," in *Concurrency Theory (CONCUR)*, 2006.

[117] A. Parma and R. Segala, "Axiomatization of trace semantics for stochastic nondeterministic processes," in *Quantitative Evaluation of Systems (QEST)*, 2004.

[118] D. N. Jansen, H. Hermanns, and J.-P. Katoen, "A probabilistic extension of uml statecharts," in *Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT)*, 2002.

[119] L. Cheung, N. A. Lynch, R. Segala, and F. W. Vaandrager, "Switched PIOA: Parallel composition via distributed scheduling," *Theoretical Computer Science (TCS)*, 2006.

[120] R. Canetti, L. Cheung, D. K. Kaynar, M. Liskov, N. A. Lynch, O. Pereira, and R. Segala, "Analyzing security protocols using time-bounded Task-PIOAs," *Discrete Event Dynamic Systems*, 2008.

[121] S. Cattani and R. Segala, "Decision algorithms for probabilistic bisimulation," in *Concurrency Theory (CONCUR)*, 2002.

[122] L. Cheung, M. Stoelinga, and F. W. Vaandrager, "A testing scenario for probabilistic processes," *Journal of the ACM (JACM)*, 2007.

[123] B. Delahaye, J.-P. Katoen, K. G. Larsen, A. Legay, M. L. Pedersen, F. Sher, and A. Wasowski, "Abstract probabilistic automata," http://perso.bretagne.ens-cachan.fr/~delahaye/rapports/VMCAI11-long.pdf, 2010.

[124] K. Larsen, U. Nyman, and A. Wasowski, "Modal I/O Automata for interface and product line theories," in *Programming Languages and Systems (PLOS)*, 2007.

[125] N. Lynch and M. R. Tuttle, "An introduction to Input/Output automata," *CWI-quarterly*, 1989.

[126] P. Crouzen and H. Hermanns, "Aggregation ordering for massively compositional models," in *Application of Concurrency to System Design (ACSD)*, 2010.

[127] M. Hennessy and R. Milner, "Algebraic laws for nondeterminism and concurrency," *Journal of the ACM (JACM)*, 1985.

[128] "STREP COMBEST (COMponent-Based Embedded Systems design Techniques)," http://www.combest.eu/home/.

[129] "ECDAR," http://www.cs.aau.dk/~adavid/ecdar/, webpage.